Un sistema de gestión de historiales clínicos de neurología

Autores: Álvaro Alcaraz García y David Cobos García
Profesor director: Antonio Sarasa Cabezuelo

Curso académico: 2016/2017

Trabajo de fin de grado del Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Autorización de difusión

UNIVERSIDAD COMPLUTENSE MADRID

AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO FIN DE GRADO Y SU DEPÓSITO EN EL REPOSITORIO INSTITUCIONAL E-PRINTS COMPLUTENSE

Los abajo firmantes, alumno/s y tutor/es del Trabajo Fin de Grado (TFG) en el Grado en .....de la Facultad de ....., autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el Trabajo Fin de Grado (TF) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Periodo de embargo (opcional):

- ☐ 6 meses
- ☐ 12meses

TÍTULO del TFG: .....
.....
Curso académico: 20..... / 20.....

Nombre del Alumno/s:
.....
.....

Tutor/es del TFG y departamento al que pertenece:
.....
.....
.....

Firma del alumno/s

Firma del tutor/es

Agradecimientos

Es debido agradecer, como no podía ser menos en primer lugar a nuestro tutor, Antonio Sarasa Cabezuelo, por habernos dado la oportunidad de realizar este Trabajo de Fin de Grado, por todo su apoyo y por haber estado encima de nosotros, aconsejándonos, guiándonos y haciendo que todo haya sido mucho más sencillo. La mezcla de guía de camino, con la libertad que nos hadado en bastantes apartados, así como mano firme cuando ha sido necesario han desembocado en la consecución de este trabajo.

En segundo lugar y no menos importante, hay agradecer también a nuestras familias, por estar siempre a nuestro lado durante todo el camino y especialmente en este último tramo, en el cual se aproxima ya el final de la carrera.

Prólogo

No es sino paso a paso como las cosas avanzan, evolucionan y van tomando forma, y ha sido paso a paso como empezó y tomó forma este proyecto.

Al igual que como se empezó en la universidad, con más ilusiones que ideas, es como comenzó este trabajo de fin de grado. Una puesta en común de ideas, tecnologías y ambición, las cuales todas juntas han dado pie a lo conseguido actualmente.

Afrontando el que iba a ser con total seguridad el último trabajo de la carrera, dos amigos se juntaron y pusieron la maquinaria en marcha para comenzar este trabajo de fin de grado.

Tras la lluvia de ideas inicial, pronto ambos compañeros acabaron decantándose hacia el aspecto médico, sobre todo con la influencia de progenitores médicos en una de las casas.

Se buscó cómo favorecer a los especialistas aprovechando los conocimientos adquiridos en los años de carrera, programación, aplicaciones, web, bases de datos, etc. De ahí surgió la idea de un sistema de gestión de informes médicos sobre pacientes.

Ofrecer la posibilidad a un médico de mejorar su gestión y control sobre la información del paciente.

Resumen

De un tiempo a esta parte, se ha podido observar el uso creciente de las nuevas tecnologías en el ámbito sanitario en campos tales como la monitorización de pacientes, el soporte para la realización de operaciones quirúrgicas, la gestión de la información de hospitales o centros sanitarios, la gestión de expedientes médicos, etc.

En este sentido, el trabajo que se ha desarrollado en este TFG tenía como finalidad ayudar a los médicos del área de Neurología en la realización y gestión de pruebas de diagnóstico y control de enfermedades neurológicas a los pacientes. Se trata de una actividad rutinaria que realizan en las consultas y que normalmente llevan a cabo de una forma manual. El objetivo era proporcionar una aplicación que facilitará al médico poder realizar digitalmente este tipo de pruebas, generar plantillas de pruebas adaptados a cada tipo de paciente, y permitir guardar toda la información generada sobre un paciente en un perfil asociado al mismo en la aplicación (el historial digitalizado de un paciente). Además, un requisito básico era conseguir que la funcionalidad fuera accesible de manera intuitiva y simple al médico.

Una de las características claves de la aplicación desarrollada consiste en la capacidad de poder realizar los tests al paciente desde dispositivos móviles. Dado el ámbito de aplicación, la realización de las pruebas desde un ordenador de mesa hace que la interacción con el paciente sea más fría y alejada. En este sentido, la posibilidad de realizar el test desde una tablet o un móvil permite al médico acercarse al paciente, realizar la prueba e ir rellenándola en el dispositivo móvil, lo que resulta menos intrusivo en la relación médico paciente. Es por esta razón, que la aplicación consta de dos componentes diferentes. Una aplicación web que permite gestionar los pacientes, las plantillas de tipos de test, el control de documentos, etc., y una aplicación móvil orientada exclusivamente a la realización de test a los pacientes.

Abstract

During a long time, it has been possible to observe the increasing use of the modern technologies in the health treatment, in fields such as patient monitoring, support for surgical operations, management of hospitals or health centers, management of clinical records, etc.

On this way, the work that has been developed in this TFG was aimed at helping physicians in the area of neurology in the implementation and management of diagnostic tests and control of neurological diseases to patients. It is a routine activity that the physicians perform in the consultations.

The objective was to provide an application that will enable the physicians to digitally perform this type of tests, generate test templates adapted to each type of patient, and allow to store all information generated on a patient in a profile. (Digitalized history of a patient). In addition, a basic requirement was to make the functionality intuitively and simply accessible to the physician.

One of the key features of the application developed is the ability to perform the tests to the patient from mobile devices. Given the scope of application, testing from a desktop computer makes the interaction with the patient cooler.

The possibility of performing the test from a tablet or a mobile allows the physician to approach the patient, perform the test and fill it in the mobile device, which is less intrusive in the patient-doctor relationship. It`s for this reason, that the application is formed by two different components. A web application that allows manage patients, templates of test, document control, etc. and a mobile application oriented exclusively to the test for patients.

Keywords

Android application, Web application, patient, medical, physician, template, MongoDB

Índice de figuras

Índice

Introducción

Objetivos

El objetivo principal de este trabajo consiste en proporcionar una herramienta a los médicos de neurología para gestionar digitalmente los historiales clínicos de sus pacientes. Un historial clínico está formado por diversos tests a los que son sometidos los pacientes para predecir y controlar enfermedades neurológicas. La herramienta debería permitir a cualquier médico crear plantillas de tests personalizadas a cada tipo de paciente, gestionar los historiales clínicos de sus pacientes, y acceder a las estadísticas de los test realizados.

Los objetivos específicos planteados en el proyecto son los siguientes:

- Desarrollar una aplicación web que permita a un médico crear historiales clínicos digitales de pacientes con una estructura definida por defecto o bien a partir de plantillas personalizadas definidas por el propio médico. Así mismo, el médico podrá crear tests de diagnóstico de enfermedades que se asociarán a los historiales clínicos de cada paciente. Los test serán creados a partir de plantillas definidas por los médicos. Por último, la aplicación web ofrecerá la generación de diversos tipos de estadísticas a partir de los datos recogidos en los historiales clínicos tales como el análisis de las enfermedades por rangos de edades o sexo de los pacientes a los que se han realizado test.
- Diseñar una aplicación móvil orientada a la realización de los tests de diagnóstico a los pacientes que se han definido a través de la aplicación web. Los resultados de los tests, serán almacenados en tiempo real en los historiales clínicos de los pacientes.

Estado del arte

La gestión de los pacientes por parte de clínicas, hospitales y centros de salud es de gran importancia para mejorar el trato que se ofrece a dichos pacientes, así como, favorecer, de manera sustancial, a los médicos a la hora de desempeñar su trabajo. Esto permite una mejor atención, y un mejor control de pacientes, enfermedades y estadísticas.

Existe una cantidad importante de programas informáticos que se dedican a realizar esta tarea de una forma eficiente, posibilitando la generación de una base de datos centralizada, un control para los médicos y el diseño de estrategias para la solución de problemas del paciente.

Como en todos los casos, existen diferentes calidades de programas y opciones para adaptar los mismos a las necesidades de los centros de salud, hospitales o clínicas.

A continuación, se exponen una lista de programas con los mismos o similares objetivos que este trabajo:

Se va a dividir el estudio de las aplicaciones actuales en función del campo de trabajo en la que están especializadas:

- Herramientas de visión de imágenes médicas:
    - RadiAnt: [14] Se trata de un software instalable que permite visualizar cualquier tipo de imágenes médicas. Permitiendo control total sobre el tratamiento de las imágenes, zoom, brillo, rotación, etc.
    - OsirisX [15]: Se trata de un software de visor de imágenes. Ofrece técnicas de post-procesamiento avanzado en 2D y 3D.

- Herramientas de gestión de historiales clínicos
    - Mediconta: [16] Se trata de un programa médico para la gestión de clínicas. Ofrece:
        - Funciones administrativas, creación de usuarios en la aplicación y generación de copias de seguridad cumpliendo la LOPD.
        - Gestión de pacientes: creación de historias, diagnósticos, plantillas, recetas y facturación electrónica.
        - Contabilidad de la clínica.
        - Herramienta de agenda y avisos por sms e email.
        - Base de datos en la nube

    - Doctorgest: [17] es un software flexible para la gestión de todo tipo de consultas médicas. Permite:
        - Historia clínica.
        - Agenda Web, envío de sms e email.
        - Multiespecialidad
        - Control de tiempos de espera de pacientes.
        - Etiquetas a pacientes, control de datos y cumplimiento de la LOPD.

- Herramientas de gestión de citas médicas
    - Jagarsoft: [18] software de agenda multiusuario optimizada.
        - Permite búsquedas rápidas de pacientes y asignación de citas recurrentes.
        - El apartado de citas permite un control de reportes y estadísticas.
        - Información en tiempo real y recordatorio de citas, así como aviso de nuevas citas y cancelaciones.
    - ClinicCloud: [19] software para la gestión de citas médicas para consultorios.
        - Permite selección de días de citas, hora y al profesional que la va a llevar a cabo.
        - Permite control de datos personales de los pacientes.
        - Notificaciones por correo en caso de aceptar o cancelar una cita.
        - Evita solapamientos de horas entre clientes y médicos.

Plan de trabajo

En esta sección se va a explicar brevemente la estructura de la memoria:

- **Sección 1: Especificación de la aplicación**

  En esta sección se explican las distintas funcionalidades de la aplicación, los actores que intervienen y los casos de uso que se han definido.

- **Sección 2: Tecnologías utilizadas**

  En esta sección se exponen las tecnologías que han sido utilizadas para desarrollar el sistema.

- **Sección 3: Arquitectura de la aplicación**

  En esta sección se detallan los componentes software que constituyen el sistema y como se encuentran relacionados.

- **Sección 4: Modelo de datos**

  En esta sección, se describe la estructura de la base datos y las entidades de las que se guarda información.

- **Sección 5: Diseño**

  En esta sección, se explica la parte relativa al diseño de la aplicación. En particular se describe las características de la interface de usuario.

- **Sección 6: Implementación**

  En esta sección, se detalla los aspectos relacionados con la codificación del sistema.

- **Conclusiones y trabajo futuro**

  En este apartado se establecen las conclusiones y las posibles líneas de trabajo futuro.

Introduction

Objectives

The main objective of this work is to provide a tool for physicians of neurology to digitally manage the clinical records of their patients.

A clinical record consists of several tests that patients undergo to predict and control neurological diseases. The tool should allow any physician to create customized test templates for each type of patient, manage the clinical record of their patients and access the statistics of the test performed.

The specific objectives of the project are:

- Develop a web application that allows a physician to create digital clinical records of patients with a structure defined by default, or from personalized templates defined by the physicians. Also, the doctor can create diagnostic tests for diseases that will be associated with the clinical records of each patient. The test will be created from templates defined by doctors. Finally, the web application will provide the generation of several types of statistics from data collected in clinical records such as the analysis of diseases by age or sex ranges of patients who have been tested.
- Design a mobile application oriented to the realization of the diagnostic test to the patients that have been defined through the web application. The results of the tests will be stored in real time in the patients` medical records.

State of the art

The management of patients by clinics, hospitals and health centers is of great importance to improve the treatment offered to such patients, as well as substantially supplying physicians in the performance of their work. This allows better care, and better control of patients, diseases and statistics.

There is an important amount of computer programs that are dedicated to performing this task in an efficient way, allowing the generation of a centralized database, a control for physicians and the design of strategies for solving patient problems.

As in all cases, there are different qualities of programs and options to adapt them to the needs of health centers, hospitals or clinics.

The study of current applications will be divided according to the field of work in which they are specialized

- Medical imaging tools:
    - Radiant: [14] it is an installable software that allows to visualize any type of medical images. Allowing full control over the treatment of images, zoom, brightness, rotation, etc.
    - OsirisX : [15] This is an image viewer softer. It offers advanced post-processing techniques in 2D and 3D.

- Clinical records management tools:
    - Mediconta: (http://www.infonetsoftware.com/) this is a medical program for the management of clinics. Offers:
        - Administrative functions, user creations in the application, and backup generation according to LOPD.
        - Patient management, creation of clinical records, diagnoses, templates, recipes and electronic invoicing.
        - Clinic accounting.
        - Calendar option, a message tool by sms and email.
        - Database in the cloud.

    - Doctorgest: (http://www.doctorgest.com/) is a flexible software for the management of all types of medical consultations. It allows:
        - Clinical records.
        - A web agenda, sending sms and emails.
        - Multispecialty.
        - Control waiting patient time.
        - Patient labels, data control and LOPD information control.

- Clinical appointment management tools:
    - Jagarsoft: [18] optimized multiuser calerndar software.
        - Allows fast patient searches and assignment of recurring appointments
        - The appointments section allows the control of reports and statistics.
        - Real-time information and appointment reminder, as well as notices of new appointment or cancellations.
    - ClinicCloud: [19] office management software for clinics.
        - Allows selection of dates of appointment and the professional that will carry out.
        - Allows control of patients` personal data
        - Notifications by email in case of accepting or cancelling an appointment.
        - Avoid time overlaps between clients and doctors.

Working Schedule

In this section, we will briefly explain how the memory and the evolution of the project are structured.

- **Section 1: Application specification**

  This section explains the various functionalities of the application, the actors involved and the use cases that can be presented

- **Section 2: Technologies used**

  This section discusses the different technologies used for both web and mobile applications.

- **Section 3: Application architecture**

  The different modules that make up the work are detailed, both the web part and the mobile part

- **Section 4: Data model**

  The structure of the database is detailed, explaining the collections of which is formed and its use.

- **Section 5: Design**

  This section explains everything related to the design in the application. Focused mainly on the interface.

- **Section 6: Implementation.**

  It explains the methods used to carry out the writing of the code that conforms the application, focusing on the use cases.

- **Conclusions and future work**

  Final part of the work, where the test made after project ended. The conclusion obtained and the bibliography used.

Proyecto

Sección 1: Especificación de la aplicación

1.1 Funcionalidades

La funcionalidad del sistema se encuentra repartida entre una aplicación web y una aplicación móvil.

La funcionalidad de la aplicación web está enfocada a los médicos y a su trato y control de pacientes. Se permitirá al médico introducir pacientes al sistema por dos vías, mediante una plantilla por defecto, o utilizando una plantilla previamente personalizada. A estos pacientes guardados en el sistema, el médico podrá modificarlos, añadirles nuevos documentos relacionados, o eliminarlos del sistema.

Se podrán, además, crear plantillas de test de cara a su uso en la aplicación móvil.

Se ha incluido un sistema de control de estadísticas sobre los test realizados a los pacientes. Dichas estadísticas podrán generarse desde su apartado correspondiente en la aplicación y generarán gráficos representativos en función de los campos de búsqueda seleccionados.

Por último, en la aplicación web, el médico dispondrá de un perfil personal, el cual podrá modificar si así lo cree oportuno.

Con respecto a la funcionalidad de la aplicación móvil, está enfocada a la generación de test a los pacientes. El médico podrá buscar al paciente que quiera, obteniendo toda su información en la pantalla del dispositivo. Y además, el médico podrá realizarle un test. Para ello tendrá a su disposición como funcionalidad añadida la muestra de todos los tipos de plantillas de test que el médico haya creado. Por último, una vez elegido paciente, y elegida plantilla, se realiza el test.

1.2 Actores de la aplicación

En la aplicación web se han identificados los siguientes actores:

- Administrador: dispone de una cuenta de superusuario, por lo que simplemente al comienzo se identificará como tal. Una vez hecho el log-in se le carga la vista de administración en la cual podrá realizar las siguientes acciones:
  - Búsqueda y actualización de perfiles: el administrador será capaz de ver los datos personales buscándolos por DNI en caso de necesitarlo podrá modificar algún dato personal.
- Médico: Tendrá dos opciones al acceder a la interfaz de bienvenida, si ya dispone de cuenta, y está confirmada, introducirá sus datos y entrará. Si no está confirmada la cuenta le saldrá un cartel informativo, y si no dispone de una, tendrá la oportunidad de registrarse rellenando un formulario y pasará al punto de espera de confirmación.
  - Dar de alta un paciente: dispondrá de un botón para dar de alta, el cual le permitirá elegir entre una plantilla por defecto o una de las plantillas previamente creadas.
  - Dar de baja a un paciente: se realizará mediante la opción de búsqueda por DNI del paciente.
  - Modificar/ver datos de un paciente: también aprovechando la utilidad de la búsqueda, mediante un botón se podrá acceder al historial y modificar los datos pertinentes.

- Exportar los datos personales: tras la búsqueda del paciente se podrá exportar su historial a pdf.
- Exportar documentos adjuntos del paciente: al igual que con la descarga de datos personales, existe la opción de descargar todos los documentos que estén adjuntos al mismo, en un fichero de formato ".zip".
- Crear plantilla personalizada de registro: por defecto, y de cara a estadísticas, contendrá unos campos inamovibles, y a partir de ahí podrá el médico añadir y eliminar campos (de cualquier tipo, texto plano, combobox, etc.) a su gusto. Dispondrá también la opción de marcar una de las plantillas para que sea la usada por defecto.
- Crear plantilla personalizada de test o exploraciones: se creará un formulario mediante la agregación de campos de tipo texto.
- Generar estadísticas: se generarán estadísticas en función a valores elegidos entre rango de edades, sexo y tipo de enfermedad.

Con respecto a la aplicación Android, solo se ha identificado el actor "Médico". Este actor actúa desde la pantalla de log-in, para lo cual debe haberse registrado previamente en la web y haber sido aceptado por el administrador (no se puede registrar desde la aplicación móvil). La funcionalidad principal de la App para el médico va a ser el poder añadir varios test a la vez que se realiza la consulta. Así mismo puede:

- Buscar el historial del paciente: mediante el nombre/nº historial.
- Añadir test: desde el historial se podrá añadir un nuevo test al paciente. Este test va a poder ser un comentario mediante texto plano, o bien, seleccionar una de las plantillas personalizadas de test, que han sido creadas previamente desde la AW.

1.3 Casos de uso

Se van a presentar los casos de uso de acuerdo a si afectan a la aplicación web o si afectan a la aplicación móvil.

a) Aplicación web

Los casos de uso se refieren a los actores Administrador y Médico. Con respecto al Administrador, se asumirá en todos los casos de uso (Figura 1.3.1) que está autenticado correctamente.

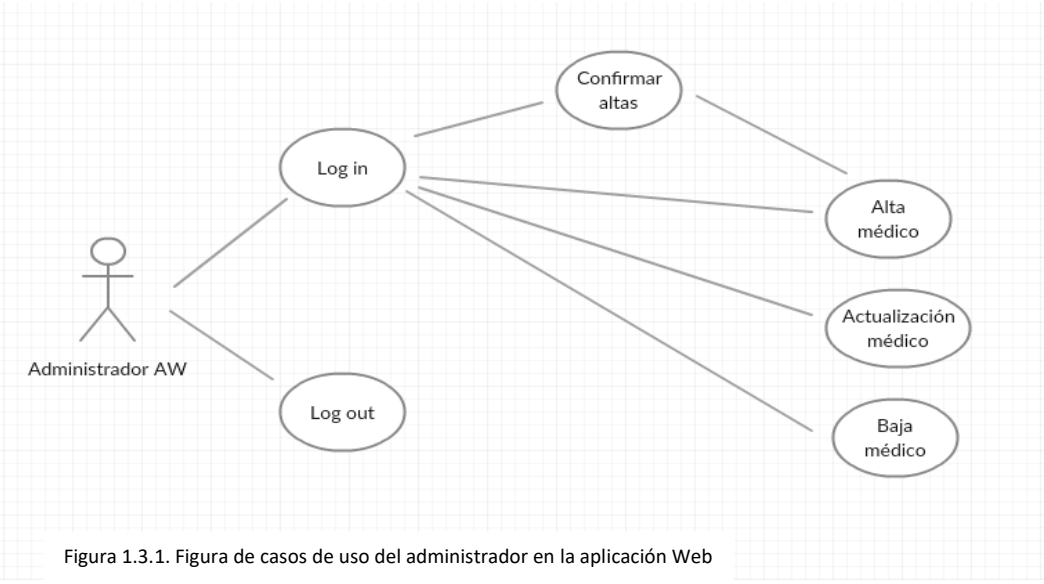

Figura 1.3.1. Figura de casos de uso del administrador en la aplicación Web

A continuación, se listan los casos de uso.

Caso de uso	**Dar de alta un médico**
Objetivo de contexto	Insertar un nuevo médico a la lista de médicos existentes.
Precondiciones	El médico a insertar no debe constar previamente en la base de datos. En un futuro se validarán los datos, tanto nombre y apellidos como número de colegiado, frente a una base de datos externa.
Final exitoso	Se inserta correctamente.
Final fallido	Se repite la petición de inserción puesto que algún dato no es correcto. De nuevo esto se realizará en actualizaciones futuras.
Actores principales	El administrador de la aplicación.
Evento de inicio	Siendo administrador, se seleccionará el botón “Dar de alta” en la interfaz.
Flujo principal	• Seleccionará la opción de “Dar de alta un médico”. Pasará a una pantalla en la cual introducirá los datos del médico. • Y dará a “Dar de alta”.
Caso de uso	**Confirmar un alta**
Objetivo de contexto	Aceptar la solicitud de un médico que desea inscribirse en el sistema.
Precondiciones	Es necesario que un médico haya rellenado la solicitud de inscripción al sistema.
Final exitoso	El médico queda añadido en la base de datos de médicos.
Final fallido	El administrador rechaza la inscripción del médico.
Actores principales	Administrador y médico.
Evento de inicio	Relleno de la solicitud de inscripción por parte del médico.
Flujo principal	• En este caso tendrá un icono de notificaciones que indicará si tiene peticiones de aceptación en el sistema. • Al pulsar saldrá una lista con los médicos solicitantes. • Aceptará a aquellos que cumplan una serie de requisitos.

Caso de uso	**Dar de baja un médico**
Objetivo de contexto	Eliminar un médico existente de la aplicación.
Precondiciones	El médico debe de existir y haber pedido su baja con antelación.
Final exitoso	Se elimina correctamente.
Final fallido	No se eliminará al médico y se avisará al administrador con el error correspondiente.
Actores principales	Administrador y médico.
Evento de inicio	Llegará la notificación de la petición de baja al panel web del administrador.
Flujo principal	• El médico se encargará de notificar al administrador su baja y el administrador de la aplicación lo eliminará gracias al botón “Dar de baja”.
Caso de uso	**Buscar y actualizar un médico**
Objetivo de contexto	Buscar y actualizar la información de un médico dentro del sistema.
Precondiciones	Que exista un médico con el campo de búsqueda
Final exitoso	Se encuentra el médico y se modifican los datos que el administrador desee.
Final fallido	No se encuentra ningún médico que coincida con el campo de búsqueda.
Actores principales	Administrador
Evento de inicio	El administrador rellena el campo “Búsqueda” en su interfaz.
Flujo principal	• Se realizará la búsqueda de médico. • Se seleccionará su perfil y se podrán realizar los cambios necesarios. • Una vez modificados se tendrá la opción de guardarlos o deshacerlos.

Con respecto al Médico, si no dispone de cuenta se tendrá que registrar en una interfaz de login y esperar la confirmación por parte del administrador. Para los siguientes casos de uso (Figura 1.3.2) se asumirá que el médico está autenticado correctamente.

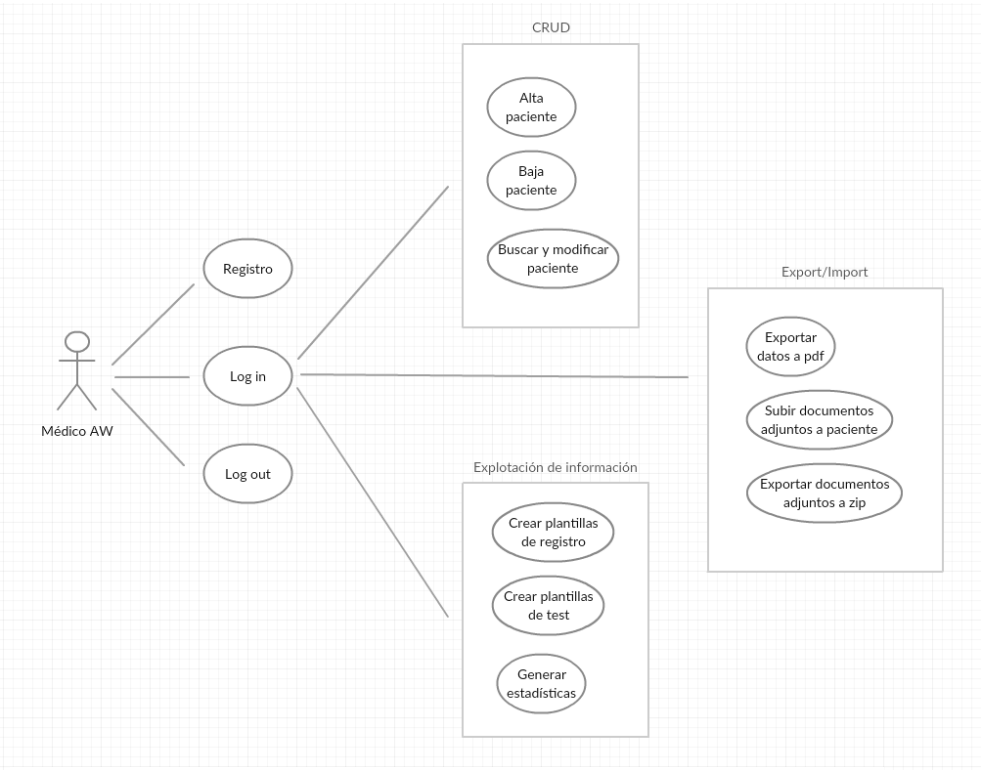

Figura 1.3.2. Figura de casos de uso del médico en la aplicación Web

Caso de uso	**Dar de alta pacientes**
Objetivo de contexto	Insertar un nuevo paciente a la lista de historiales asociados al médico.
Precondiciones	El paciente a insertar no debe constar previamente en la base de datos.
Final exitoso	El paciente se inserta correctamente.
Final fallido	No se inserta el paciente, y se le avisa al médico con el error correspondiente para que reinicie el proceso de alta.
Actores principales	Médico.
Evento de inicio	Primera visita del paciente a la consulta.
Flujo principal	• Seleccionará la opción de “Dar de alta”, reflejada con un icono con el símbolo “+”. • El médico elegirá la plantilla de historial deseada, por defecto o previamente creada. • Pasará a una pantalla en la cual introducirá los datos del paciente. • Y dará a “Aceptar”.

Caso de uso	**Dar de baja pacientes**
Objetivo de contexto	Eliminar un paciente de la lista de historiales.
Precondiciones	El paciente debe constar previamente en la base de datos.
Final exitoso	El paciente se elimina correctamente.
Final fallido	El paciente no se elimina correctamente, y se le avisa al médico con el error correspondiente para que reinicie el proceso de alta.
Actores principales	Médico.
Evento de inicio	El paciente deja de estar asociado al médico.
Flujo principal	• Se realizará una búsqueda por número de historial. • Se seleccionará al paciente. • Se le dará de baja.
Caso de uso	**Buscar y actualizar un paciente**
Objetivo de contexto	Modificar datos sobre el historial del paciente.
Precondiciones	El paciente deberá existir en la base de datos.
Final exitoso	Se modificará el campo o campos deseados.
Final fallido	No se modifica el historial del paciente, y se avisará al médico del error.
Actores principales	Médico.
Evento de inicio	El médico rellena el campo “Búsqueda” en su interfaz.
Flujo principal	• Se realizará la búsqueda de paciente. • Se seleccionará su historial y se podrán realizar los cambios necesarios. • Una vez modificados se tendrá la opción de guardarlos o deshacerlos.

Caso de uso	**Exportar datos a pdf**
Objetivo de contexto	Generar un archivo .pdf con la información relativa a un paciente.
Precondiciones	El paciente debe existir previamente en la base de datos.
Final exitoso	Se genera un fichero .pdf con la información del paciente
Final fallido	No se genera el fichero, y se avisará al médico del error.
Actores principales	Médico.
Evento de inicio	Se buscará el historial del paciente con intención de exportar sus datos a pdf
Flujo principal	• Se buscará el historial del paciente. • Dentro del mismo se tendrá la opción de exportar a pdf.
Caso de uso	**Subir documentos adjuntos a pacientes**
Objetivo de contexto	Se subirá un documento adjunto a ese paciente.
Precondiciones	El paciente debe existir previamente en la base de datos y el archivo no puede existir antes para evitar copias.
Final exitoso	Se adjunta el documento al paciente y se genera en su carpeta.
Final fallido	No se producen cambios.
Actores principales	Médico.
Evento de inicio	Se buscará el historial del paciente con intención de subir los documentos a su perfil.
Flujo principal	• Se buscará el historial del paciente. • Dentro del mismo se tendrá la opción de subir documento. • Se adjunta el documento y se genera una copia en la carpeta del paciente.

Caso de uso	**Exportar documentos adjuntos a zip**
Objetivo de contexto	Generar un archivo .zip con los documentos relativos a un paciente.
Precondiciones	El paciente debe existir previamente en la base de datos.
Final exitoso	Se genera un fichero .zip con todos los documentos del paciente.
Final fallido	No se genera el fichero, y se avisará al médico del error.
Actores principales	Médico.
Evento de inicio	Se buscará el historial del paciente con intención de exportar sus documentos a un fichero zip
Flujo principal	• Se buscará el historial del paciente. • Dentro del mismo se tendrá la opción de descargar fichero zip. • Se descarga un fichero en la carpeta del paciente.
Caso de uso	**Crear plantillas de registro**
Objetivo de contexto	Crear plantillas de historial personalizadas con los campos específicos que considere el médico.
Precondiciones	No precisa de precondiciones.
Final exitoso	Se añadirá la plantilla creada a la lista de plantillas de historiales asociadas al médico.
Final fallido	No se añadirá la plantilla y se le notificará al médico con el error.
Actores principales	Médico.
Evento de inicio	Se pulsa en “Añadir plantilla de historial “.
Flujo principal	• Pulsar en “Añadir plantilla de historial” en la pantalla principal del médico. • Añadir campos específicos pulsando en “+”. • Confirmar la plantilla pulsando en “Aceptar”.

Caso de uso	**Crear plantillas de registro**
Objetivo de contexto	Crear plantillas de historial personalizadas con los campos específicos que considere el médico.
Precondiciones	No precisa de precondiciones.
Final exitoso	Se añadirá la plantilla creada a la lista de plantillas de historiales asociadas al médico.
Final fallido	No se añadirá la plantilla y se le notificará al médico con el error.
Actores principales	Médico.
Evento de inicio	Se pulsa en “Añadir plantilla de historial “.
Flujo principal	• Pulsar en “Añadir plantilla de historial” en la pantalla principal del médico. • Añadir campos específicos pulsando en “+”. • Confirmar la plantilla pulsando en “Aceptar”.
Caso de uso	**Crear plantillas de test**
Objetivo de contexto	Crear plantillas de test personalizadas con los campos específicos que considere el médico.
Precondiciones	No precisa de precondiciones.
Final exitoso	Se añadirá la plantilla creada a la lista de plantillas de test asociadas al médico.
Final fallido	No se añadirá la plantilla de test, y se le notificará al médico con el error.
Actores principales	Médico.
Evento de inicio	Pulsar en “Añadir plantilla de test”.
Flujo principal	• Pulsar en añadir plantillas de test, • Añadir campos específicos pulsando en “+” • Confirmar la plantilla de test pulsando en “Aceptar”.

Caso de uso	**Generar estadísticas**
Objetivo de contexto	En función de unos parámetros generar unas gráficas representativas de cada estadística buscada.
Precondiciones	Deben existir test adjuntos a pacientes. Debe existir algún valor en los inputs para generar dichas gráficas.
Final exitoso	Se genera la gráfica con la estadística buscada.
Final fallido	No se genera la gráfica con la estadística buscada.
Actores principales	Médico.
Evento de inicio	El médico rellena uno o varios campos en el formulario de valores buscados para estadística.
Flujo principal	• Se insertarán valores en función de la estadística buscada. • Se pulsará en el botón “Crear estadística”.

b) Aplicación Android.

Todos los casos de uso están referidos al Médico (Figura 1.3.3) y en todos ellos se va asumir que el médico está identificado correctamente.

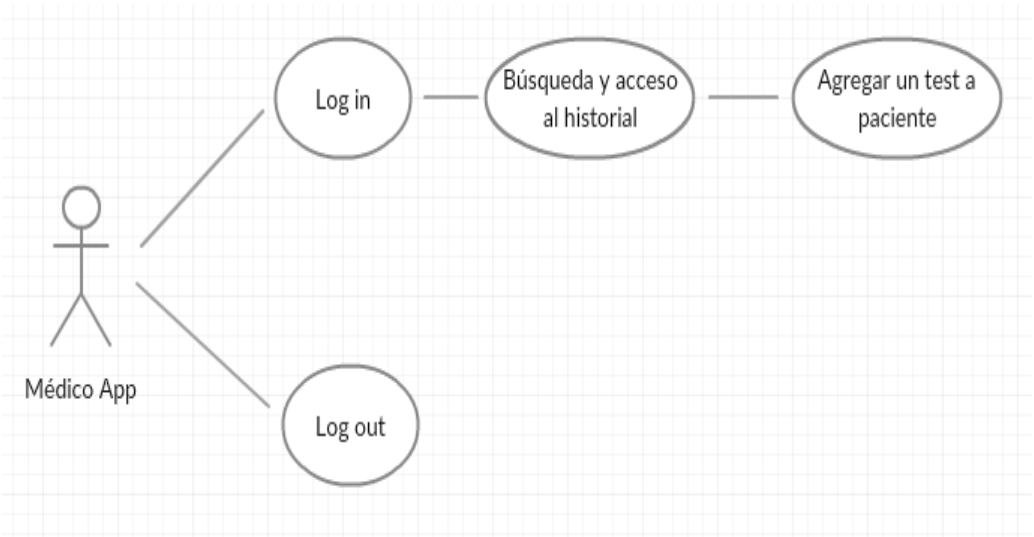

Figura 1.3.3. Figura de casos de uso del médico en la aplicación Android

Caso de uso	**Búsqueda y acceso al historial**
Objetivo de contexto	Obtener en la pantalla de la aplicación la información de un paciente.
Precondiciones	El paciente debe existir en la base de datos de historiales.
Final exitoso	Se visualiza correctamente por pantalla la información del paciente.
Final fallido	No se visualiza la información del paciente, y se informa al médico del error.
Actores principales	Médico.
Evento de inicio	El médico rellena el campo “Búsqueda” en la interfaz de la aplicación.
Flujo principal	• Se buscará al paciente. • Se mostrará su información por pantalla
Caso de uso	**Agregar un test a un paciente**
Objetivo de contexto	Añadir un test, eligiendo una plantilla previamente creada, al historial del paciente.
Precondiciones	El historial del paciente debe de existir y el test debe estar rellenado.
Final exitoso	Se añade el test al historial del paciente.
Final fallido	No se añade el test al historial del paciente y se informa al médico del error.
Actores principales	Médico.
Evento de inicio	Buscar el paciente con intención de agregarle un nuevo test.
Flujo principal	• Se buscará al paciente. • Se pulsará el botón “Elige plantilla”. • Se elegirá el test requerido. • Se aceptará pulsando “Finalizar” una vez rellenado el test.

Sección 2: Tecnologías utilizadas

2.1 Aplicación Web

Para implementar la parte del FrontEnd se ha escogido como tecnología de marcado, HTML5, por ser la más extendida y permitir la inclusión de contenido multimedia gracias a sus etiquetas con codecs, generar tablas dinámicas para manejar grandes conjuntos de datos y mejoras en los formularios para que no sea necesario validar con javascript. Como framework se ha utilizado Bootstrap[2], ya que ofrece las facilidades necesarias para desarrollar una interfaz que se comporte de manera full "responsive", es decir, adaptándose dinámicamente al tamaño de pantalla donde se muestra.

HTML5, es un estándar que sirve para la elaboración de páginas web. Es un lenguaje tipado, controlado por etiquetas y que permite su ejecución en cualquier navegador [7].

Bootstrap es un framework para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, botones, cuadros y otros elementos de diseño basados en HTML y CSS, de ahí que se haya seleccionado como framework para esta aplicación. Para aprender y utilizar Bootstrap [3]

En cuanto a la parte visual, CSS3 ha sido la tecnología elegida, ya que ofrece facilidades para separar la estructura del documento de su presentación respetando compatibilidades en los navegadores.

CSS3 es un lenguaje de diseño gráfico para definir el formato del lenguaje HTML [21].

Para un mejor uso de este lenguaje se buscaron ejemplos de uso en la página https://www.w3schools.com/css/

Para aumentar la experiencia de usuario y la usabilidad de las aplicaciones, se han utilizado fuentes vectoriales, ya que permiten su fácil introducción y modificación, y liberan a la página web de la carga de imágenes pesadas. Se ha elegido FontAwesome por su amplio abanico de fuentes.

FontAwesome es una herramienta de control y uso de fuentes e iconos, basada en CSS [4]

Y también se ha utilizado Javascript, porque aporta dinamismo en el uso de las páginas web en el lado del cliente. Además, se han utilizado librerías como Chart.js, [5] que permiten mostrar gráficas embebidas en un html, pudiendo cargar datos directamente de consultas de bases de datos, tanto relacionales como no relacionales, y aplicarles estilos de manera sencilla.

Con respecto al Backend, se ha utilizado PHP como lenguaje del lado del servidor, ya que se complementa perfectamente con Javascript, sobre todo para webs de contenido dinámico como es el caso. PHP es un lenguaje de programación diseñado para el desarrollo web.

Además, como método de exportación de pdf, se ha elegido TCPdf, ya que, al ser una clase escrita para PHP, permite su fácil adaptación en la arquitectura de proyecto en cuestión. TCPdf es una librería open source para PHP que permite crear PDF interpretando además código XHTML.

2.2 Aplicación Android

La tecnología elegida para el desarrollo de la aplicación móvil ha sido Android, ya que al ser código abierto, dispone de una comunidad más amplia de desarrolladores, y hay un mayor abanico de entornos con los que poder trabajar. A propósito de esto, la plataforma utilizada para esta parte del proyecto ha sido Android Studio [11].

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android, reemplazando a Eclipse como IDE oficial de desarrollo de aplicaciones para Android.

- Para el FrontEnd
  Se utiliza la declaración de elementos xml en la UI, más conocidos como layouts, que ofrecen la creación de elementos para la construcción de las pantallas de la aplicación móvil.

- Para el BackEnd
  La tecnología que permite y está enfocado Android Studio es Java, por lo que todas las clases creadas en la aplicación móvil están en este lenguaje de programación.

Para la instalación del IDE Android Studio se ha utilizado la página: https://developer.android.com/studio/index.html?hl=es-419

Para el aprendizaje y manejo de Android se ha recurrido a la siguiente página: http://www.android-ide.com/tutorials.html.

2.3 Base de datos

Debido a la naturaleza del proyecto, la tecnología utilizada para el diseño de la base de datos ha sido MongoDB, ya que aporta mayor flexibilidad en cuanto a la estructura de los datos almacenados, y mayor facilidad a la hora de su tratamiento cuando se genera gran cantidad de información.

MongoDB es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida [10]

En este caso MongoDB era la opción perfecta, ya que habitualmente en medicina las bases de datos son enormes, y no siempre se rigen por un mismo formato o estructura.

Es decir, los campos de las tablas de la base datos serán variables, puesto que las plantillas serán personalizadas con tantos campos como el médico quiera. Realizar ese modelo con una base datos relación nos pareció altamente costo en comparación con MongoDB.

Sección 3: Arquitectura de la aplicación

La arquitectura de la aplicación está formada por dos aplicaciones claramente diferenciadas: una web y una aplicación Android, relacionadas a través de un sistema de persistencia compartida (Figura 3.1) implementado mediante una base datos MongoDB.

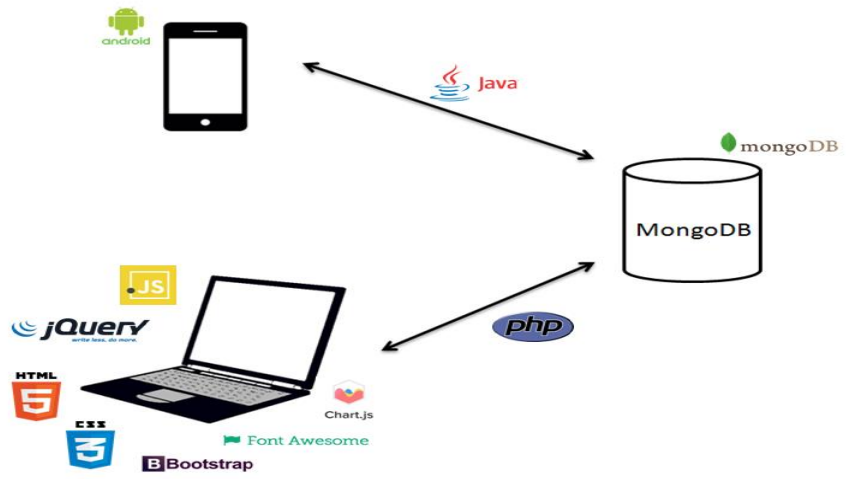

Figura 3.1. Figura de la arquitectura de la aplicación

3.1 Una aplicación web

Ofrece una interfaz amigable, a la que podrán acceder el administrador para las tareas de control y el médico para gestionar los historiales de sus pacientes y los test de exploración.

Se ha buscado separar en la medida de lo posible la interfaz gráfica del sistema de gestión de datos, construyendo para ello un modelo vista-controlador. La vista estaría formada por las páginas html. Dichas páginas se construyen dinámicamente mediante php. El controlador lo forman los scripts de php, que implementan toda la lógica de la aplicación y que servirán de conexión entre la vista y el modelo de datos, el cual se encuentra implementado en una base de datos MongoDB.

3.2 Una aplicación Android

Con esta App, el médico tendrá la posibilidad de buscar el historial de cualquiera de sus pacientes y añadirle cualquier test que se realice en las consultas. Dentro de la aplicación Android el modelo vista controlador es más evidente y sencillo de implementar puesto el propio leguaje induce a ello. Toda la parte de la vista estará constituida por los documentos en formato “xml”, los cuales únicamente realizan la generación de la interfaz y se encargan de realizar llamadas a los documentos “java”, los cuales llevan a cabo las operaciones y ejercerán de “controlador”.

3.3 Una base de datos

Sirve de nexo entre la AW y la App, manteniendo todo el contenido actualizado de forma coherente. La base de datos, al encontrarse en un servidor remoto, permitirá múltiples conexiones simultáneas. Esta será la funcionalidad clave de la arquitectura de la base de datos, puesto que se pueden sincronizar tanto la aplicación Web como la aplicación móvil gestionando el mismo repositorio de información.

Sección 4: Modelo de datos

En esta sección, se describe el modelo de datos, el cual se basa en un conjunto de colecciones de una base de datos MongoDB:

- "peticionAlta" (Figura 4.1): esta colección almacenará todos los usuarios (médicos) que han realizado el registro a la aplicación de forma correcta y están pendientes de confirmar por parte del administrador. Consta de los mismos campos que la colección "usuarios" que se explicará a continuación, pero separarlo en dos colecciones fue decisión de diseño. Dichos campos son:
  - DNI: se utilizará este campo como motor de búsqueda
  - Número de colegiado
  - 1er Apellido
  - 2º Apellido
  - Nombre
  - Teléfono
  - Dirección
  - Especialidad Médica
  - Hospital
  - Tipo: este campo es privado de cara a la aplicación y sirve para saber si el usuario es de tipo médico o de tipo administrador.
  - Contraseña

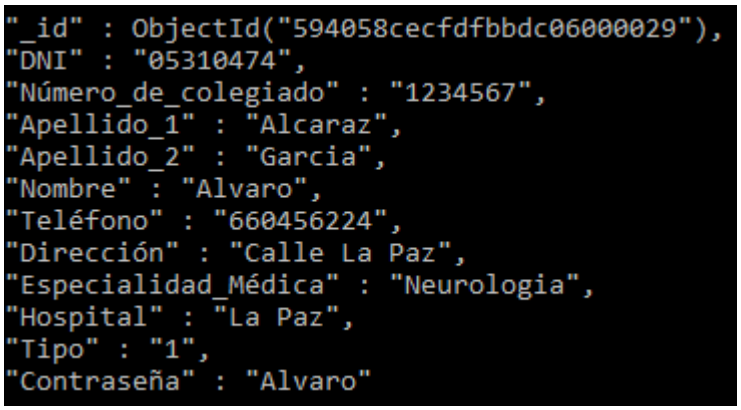

Figura 4.1. Elemento de la colección peticiónAlta

- "usuarios" (Figura 4.2): en esta colección, cada médico estará representado por un documento independiente, con el mismo número y tipo de campos, a saber:
  - DNI: se utilizará este campo como motor de búsqueda
  - Número de colegiado
  - 1er Apellido
  - 2º Apellido
  - Nombre
  - Teléfono
  - Dirección
  - Especialidad Médica
  - Hospital
  - Tipo: este campo es privado de cara a la aplicación y sirve para saber si el usuario es de tipo médico o de tipo administrador.
  - Contraseña

```
"_id" : ObjectId("59405a0ecfdfbbdc0600002a"),
"DNI" : "05310474",
"Número_de_colegiado" : "1234567",
"Apellido_1" : "Alcaraz",
"Apellido_2" : "Garcia",
"Nombre" : "Alvaro",
"Teléfono" : "660456224",
"Dirección" : "Calle La Paz",
"Especialidad_Médica" : "Neurologia",
"Hospital" : "La Paz",
"Tipo" : "1",
"Contraseña" : "Alvaro"
```


Figura 4.2. Elemento de la colección usuarios.

- "pacientes" (Figura 4.3): esta colección es similar a la anterior, solo que se encarga de almacenar las fichas de los pacientes que el médico vaya registrando en su consulta. Los campos son fijos y son los siguientes:
  - Médico: este campo contiene el DNI del médico. Es la manera que tiene nuestro modelo de datos de relacionar los médicos con sus pacientes.
  - Número de la Seguridad Social (Nº SS)
  - Primer apellido
  - Segundo apellido
  - Nombre
  - DNI: éste sí corresponde al DNI del paciente.
  - Sexo
  - Fecha de nacimiento (en formato dd/mm/aaaa)
  - Dirección
  - Población
  - Provincia
  - Código postal
  - Teléfono


```
"_id" : ObjectId("59405abdcfdfbbdc0600002b"),
"Médico" : "05310474",
"SS" : "1234568",
"Apellido1" : "Perez",
"Apellido2" : "Sanchez",
"Nombre" : "Pedro",
"DNI" : "01234567",
"Sexo" : "H",
"Fecha_de_nacimiento" : "10/07/1992",
"Dirección" : "Paseo de la Castellana",
"Poblacion" : "Madrid",
"Provincia" : "Madrid",
"CP" : "28027",
"Teléfono" : "666222333"
```


Figura 4.3. Elemento de la colección pacientes

Además, podrán generarse campos extra, debido a la inclusión de plantillas de pacientes personalizadas. Es decir, los campos anteriores siempre aparecerán, y si hay o no más campos dependerá de la decisión del médico en usar la plantilla por defecto, o alguna de las diseñadas por él.

- “plantilla_paciente” (Figura 4.4): esta colección se compone de documentos (cada uno de los cuales representa una plantilla de paciente personalizada creada por el médico) de varios campos numerados, que servirán para almacenar el nombre de los campos a ingresar cuando se utilice la plantilla. Es decir, de cara al usuario estos nombres almacenados aquí son “invisibles”, pero para el desarrollo, imprescindibles. La correspondencia clave-valor de estos documentos, transformará el valor en clave a la hora de introducir datos. Más adelante se incorporarán varias figuras donde se explicará más detalladamente. Sí que será necesario que el Campo 1 en este caso contenga el nombre de la plantilla creada. Después, todos los campos fijos. Y por último los campos extra.

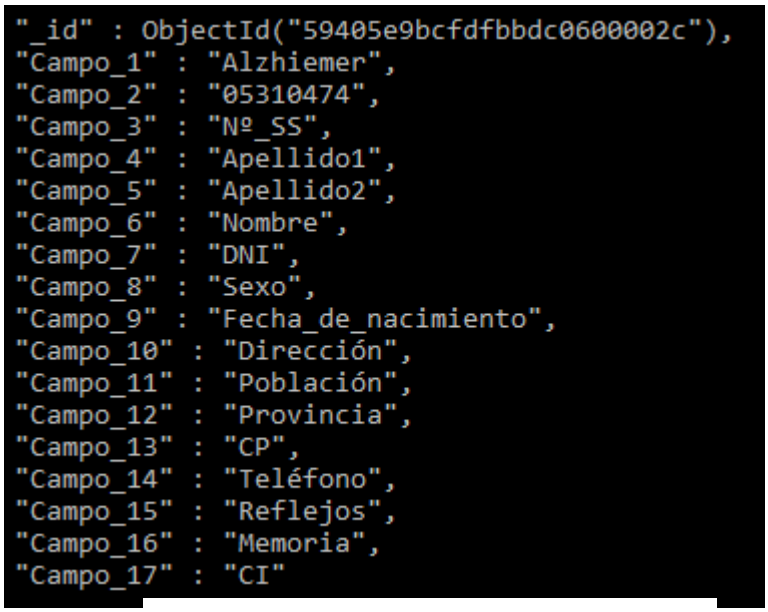

Figura 4.4. Elemento de la colección plantilla_paciente

- “plantilla_test” (Figura 4.5): esta colección funciona igual que “plantilla_paciente”, pero se utiliza para todas las plantillas de test que vaya a utilizar el médico con sus pacientes durante las consultas. Los nombres de los campos también seguirán la estructura de Campo seguida de un número. Los únicos requisitos obligatorios en cada documento son que Campo 1 contenga el nombre del test, Campo 2 por su parte contenga el DNI del médico (ya que son test personales) y Campo 3 contenga una “etiqueta”. Dicha etiqueta dará un valor unívoco al test, y se utilizará para la explotación de la información de la base de datos de test, con motivos estadísticos.

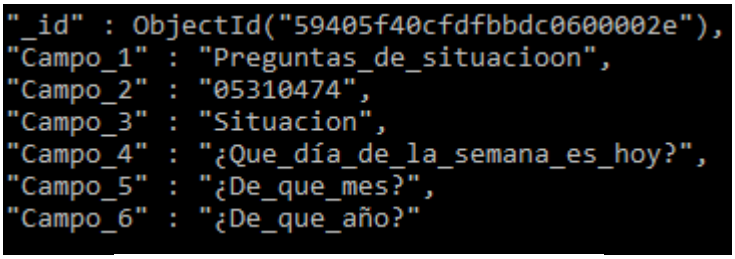

Figura 4.5. Elemento de la colección plantilla_test

- “tests” (Figura 4.6): esta colección almacenará todos los test realizados a los pacientes. Al haberse generado sus campos dinámicamente, cada test tendrá sus campos propios con sus valores, respetando los tres campos de “plantilla_test” antes descritos. También tendrán los

DNIs de médico y paciente, para hacer las referencias pertinentes en la aplicación web, desde la aplicación móvil y viceversa.


```
"_id" : ObjectId("5940605f9cbada950ddcf8be"),
"DNIPaciente" : "01234567",
"DNIMedico" : "05310474",
"Nombre" : "01234567-13-Jun-2017-21:59:59",
"¿De_que_mes?" : "Junio",
"¿Que_día_de_la_semana_es_hoy?" : "Martes",
"Etiqueta" : "Situacion",
"¿De_que_año?" : "2017"
```


Figura 4.6. Elemento de la colección test

- "tags" (Figura 4.7): esta colección almacenará todas y cada una de las etiquetas con las que se referencien los test. De tal manera que se dará la información al médico de las etiquetas ya utilizadas previamente, para poder optimizar mejor su uso. También contiene un contador de uso para cada etiqueta, con el que, de cara a posibles ampliaciones del proyecto, se puedan extraer nuevas estadísticas globalmente.


```
"_id" : ObjectId("59405f40cfdfbbdc0600002d"),
"Etiqueta" : "Situacion",
"Contador" : 1
```


Figura 4.7. Elemento de la colección tags

En el siguiente esquema (Figura 4.8) se muestra cómo se distribuyen las colecciones utilizadas en el proyecto, y la relación que existe, de forma indirecta, entre ellas.

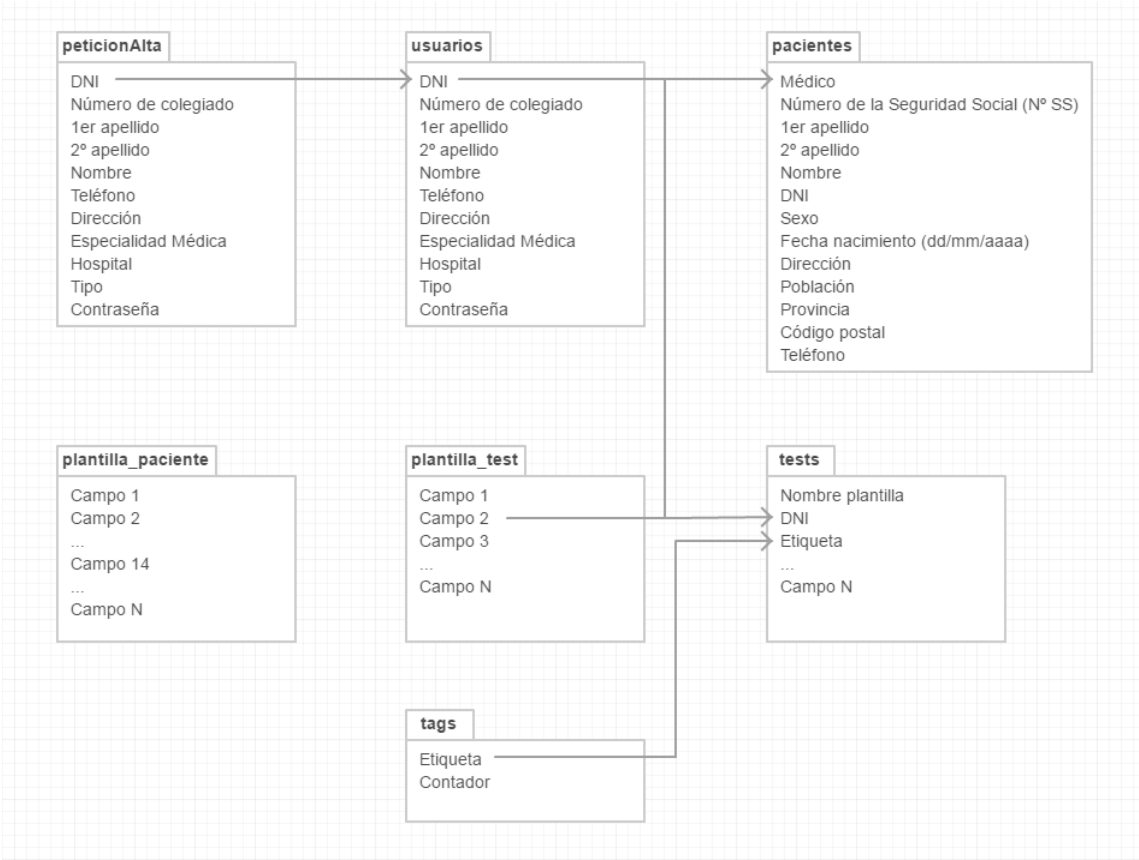

Figura 4.8. Modelo de datos

Sección 5: Diseño

5.1 En la aplicación web

A la hora de diseñar aplicaciones web y móviles, lo primero que hay que se tiene que tratar de forma explícita es, por un lado, qué ámbito de uso va a tener dicha aplicación y, por otro lado, a quién o quienes va dirigida.

El diseño se planteó para enfocar dicha aplicación a simplificar y agilizar todo el trabajo que tiene el médico en cuanto a consultas y trato con el paciente.

Al existir dos partes bien diferenciadas, web y móvil, el diseño de cada una de ellas no puede ser el mismo. A continuación, se explica el porqué de cada uno.

Para la aplicación web se optó por un diseño responsive construido sobre el framework Bootstrap. Las interfaces respetan este principio y se adaptan perfectamente a dispositivos como tablets (Figuras 5.1.1, 5.1.2 y 5.1.3).

Figura 5.1.1. Layout versión desktop

Figura 5.1.2. Layout versión tablet

Figura 5.1.3. Layout versión móvil

Se optó también por un diseño simple en bloques, que ayudase a los médicos menos familiarizados con la tecnología a interactuar de manera sencilla y rápida con la aplicación, e intentando en la mayor medida adaptarse a los principales principios de diseño web. Dichos principios son:

- Balance: se trata de equilibrar todos los elementos que componen la página, tanto texto como imágenes, zonas claras y oscuras, etc. En la interfaz diseñada se cumple este principio, ya que se dispone de un diseño simétrico en bloques. Dichos bloques constan de una parte coloreada, siguiendo patrones coherentes (verde para añadir, azul para información y rojo para borrar) y una parte de texto sin colorear que explica la funcionalidad muy brevemente. Además, todos los elementos cuentan con el mismo patrón y tamaño (Figuras 5.1.4, 5.1.5 y 5.1.6).

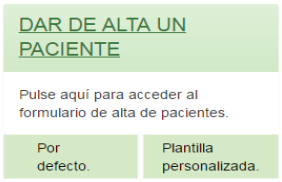

Figura 5.1.4. Estilos bloque de inserción de pacientes.

Figura 5.1.5. Estilos bloque de inserción de plantillas de pacientes.

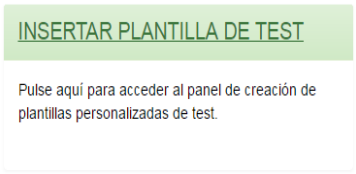

Figura 5.1.6. Estilos bloque de inserción de plantillas de test.

- Contraste: se refiere a la forma de distinguir los elementos que se quieren resaltar. Va en la línea de lo comentado en la sección de balance. Los bloques que funcionan como enlaces a funcionalidades tienen su título en color, subrayado y de mayor tamaño que la descripción breve que explica para qué sirve.

- Énfasis: es la forma de distinguir elementos. En este caso, cuando pasas el puntero del ratón por cada elemento se ilumina el bloque entero en el color que posea el título del bloque, para dejar claro que es un elemento funcional (Figura 5.1.7).

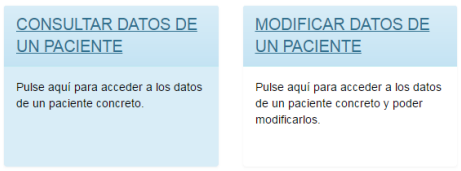

Figura 5.1.7. Estilo de bloque a seleccionar.

- Repetición o ritmo: es la forma de ordenar las cosas en la aplicación web siguiendo ciertos patrones. En este caso, se guarda cierta simetría vertical, separando los bloques en tres subsecciones que son los tres pilares de la aplicación, véase, pacientes, plantillas de pacientes, y plantillas de test (Figuras 5.1.8, 5.1.9 y 5.1.10).

Figura 5.1.8. Subsección de pacientes.

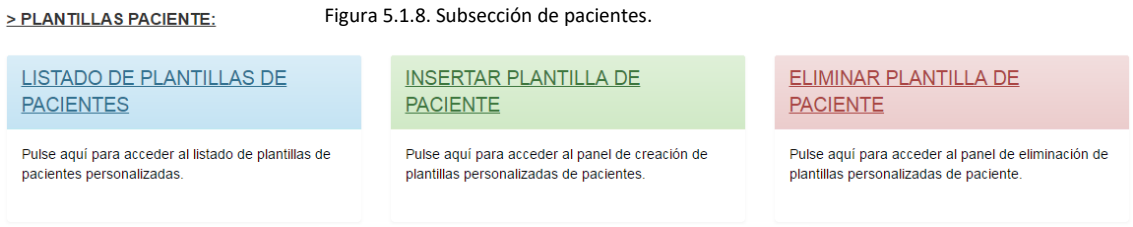

Figura 5.1.9. Subsección de plantillas de pacientes.

Figura 5.1.10. Subsección de plantilla de test.

- Proximidad o unidad: se refiere a la colocación de los elementos por temáticas guardando proximidad entre ellos. Se cumple este principio perfectamente, ya que aúna todos los puntos anteriores en este, tres secciones verticales con varios bloques horizontales, cada uno de los cuales se refiere únicamente a su subsección.

Una vez claras las estructuras y los estilos, quedaba una parte importante del diseño por fijar, en cuanto a términos de usabilidad. No es otra que los iconos a utilizar. Para que funcionase de forma óptima, con la menor carga posible se ha optado por utilizar FontAwesome, que son fuentes vectoriales que dan flexibilidad a la hora de aplicar estilos a los iconos (Figura 5.1.11), liberando así el uso de imágenes puesto que precisan cargas menos pesadas de datos.

Figura 5.1.11. Ejemplos de fuentes

5.2 En la aplicación móvil

Para la aplicación Android se optó por un diseño basado en los objetos xml “Layout” correspondientes a los ficheros xml que conforman la parte visual. Está aplicación está enfocada como se ha comentado anteriormente a un transcurso lineal, es decir, login de médico, búsqueda de paciente y realización de test al paciente.

Se comenzó con una sencilla pantalla de login (Figura 5.2.1), usuario y contraseña, donde el botón de acceder no podrá ser pulsado hasta que ambos campos no hayan sido rellenos.

La pantalla de búsqueda de paciente se ha diseñado de forma que, el médico únicamente disponga de un campo para introducir el DNI del paciente a buscar (Figura 5.2.2). También se le ofrece la opción de salir de la aplicación, para esto se ha utilizado un icono representativo de salida o finalización como es una cruz con fondo rojo.

En la pantalla de información de paciente se ha utilizado una muestra de datos en forma de lista, así como un botón para avanzar a la pantalla de selección de plantilla (Figura 5.2.3).

En la selección de plantilla, de nuevo en forma de lista, se muestra todas aquellas plantillas de test de las que dispone el médico y se le ofrece la opción de seleccionar la que desee, mediante un botón particular en cada una de las plantillas. Esta generación en forma de lista, permite delimitar claramente cada una de las posibles plantillas de test con, su título y botón correspondiente. Este diseño permite evitar fallos o errores a la hora de seleccionar una u otra plantilla (Figura 5.2.4).

Por último, el diseño de la pantalla de test es una muestra por parejas, de texto de pregunta y espacio de respuesta por cada uno de los campos de los que consta el test. De nuevo, la lista permite aislamiento de las preguntas entre sí, dejando claro cuál es el

campo de texto correspondiente para cada una y asegurarse así de que se contesta en el lugar debido (Figura 5.2.5).

Figura 5.2.1. Pantalla de login.

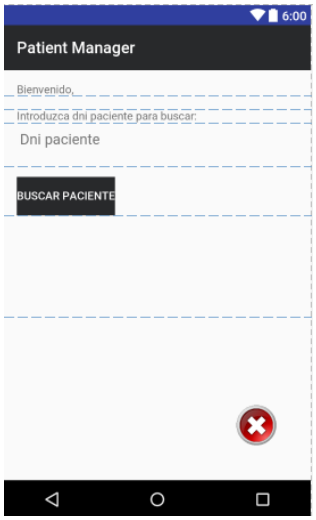

Figura 5.2.2 Pantalla de búsqueda de paciente

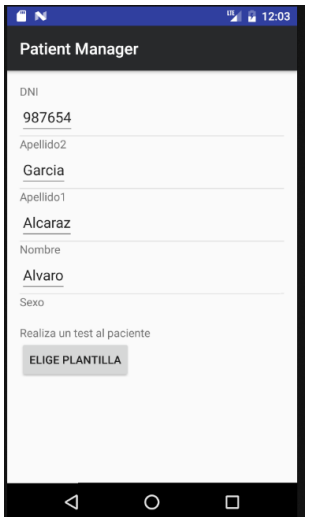

Figura 5.2.3 Pantalla de información de paciente

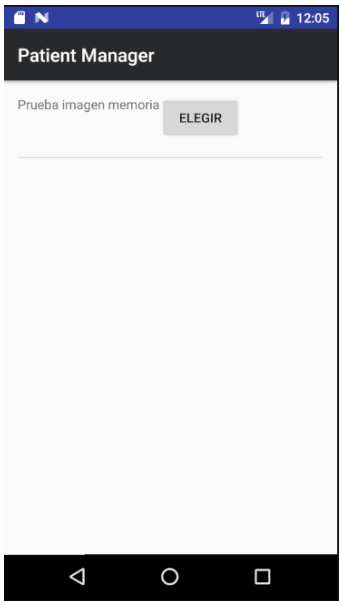

Figura 5.2.4 Pantalla de lista de plantillas de test

Figura 5.2.5 Pantalla de lista de preguntas del test

Sección 6: Implementación

6.1 Aplicación web

Esta subsección se subdivide en dos partes, una dedicada a las funcionalidades orientadas al administrador, y las orientadas al médico.

Las funcionalidades implementadas para el administrador son las siguientes:

Implementación 1. Dar de alta un médico.

Mediante esta funcionalidad, el administrador puede dar de alta de forma manual a un médico. Para ello el médico debe proporcionarle sus datos personales para introducirlos en el formulario (Figura 6.1.1).

Figura 6.1.1. Formulario de alta de médico.

A continuación, se envía el formulario con los datos vía POST al php encargado de procesarlos y almacenarlos en la base de datos. Internamente se hace un findOne con el DNI, y en caso de no estar en la base de datos se insertará. Se creará un array donde se almacenarán todos los datos cargados desde el formulario y se hará acto seguido el insert del documento que representa al médico en la colección de "usuarios", si no se produce ningún error (Figura 6.1.2).

```
$doc = array(
    "DNI"=>$dni,
    "Número_de_colegiado"=>$numCol,
    "Apellido_1"=>$apellido1,
    "Apellido_2"=>$apellido2,
    "Nombre"=>$nombre,
    "Teléfono"=>$telefono,
    "Dirección"=>$direccion,
    "Especialidad_Médica"=>$espemedica,
    "Hospital"=>$hospital,
    "Tipo" => $tipo,
    "Contraseña" => $password
);

$coll->insert($doc);
```

Figura 6.1.2. Código insert médicos.

Implementación 2. Confirmar un alta.

Esta funcionalidad sería el proceso lógico de inserción de médicos en la aplicación. Cada médico rellenará la plantilla de registro con sus datos personales, contraseña, etc. Dicho registro se realiza por medio de un formulario que pasará los datos por POST a la lógica. Y esta lógica se encargará de añadir mediante un insert en base de datos a la colección "peticionAlta", en caso de que el registro no esté pendiente, ni el médico ya forme parte de la aplicación (Figura 6.1.3).

```
$docAlta = $res->findOne(array('DNI'=>$dniconfirma),array('_id' => 0));

$doc = array(
    "DNI"=>$docAlta['DNI'],
    "Número_de_colegiado"=>$docAlta['Número_de_colegiado'],
    "Apellido_1"=>$docAlta['Apellido_1'],
    "Apellido_2"=>$docAlta['Apellido_2'],
    "Nombre"=>$docAlta['Nombre'],
    "Teléfono"=>$docAlta['Teléfono'],
    "Dirección"=>$docAlta['Dirección'],
    "Especialidad_Médica"=>$docAlta['Especialidad_Médica'],
    "Hospital"=>$docAlta['Hospital'],
    "Tipo" => $docAlta['Tipo'],
    "Contraseña" => $docAlta['Contraseña']
);


$coll->insert($doc);
$aux = $res->remove(array('DNI'=>$dniconfirma));
```

Figura 6.1.3. Código confirmar petición de alta.

En la barra de navegación existe una etiqueta o "badge" que se encarga de avisar al administrador si existen peticiones de registro pendientes por confirmar (Figura 6.1.4). Si el número que aparece es cero, no habrá ninguna petición pendiente. Si por el contrario el número es distinto de cero, significará que hay peticiones por atender. Para poder verlo con detalle, el administrador podrá pulsar sobre "Peticiones" y acceder así a la lista.

Figura 6.1.4. Badge.

El administrador entrará en este apartado y se mostrarán todos los médicos pendientes de registro de forma tabulada, con tres opciones, confirmar, ver más datos y rechazar (Figura 6.1.5).

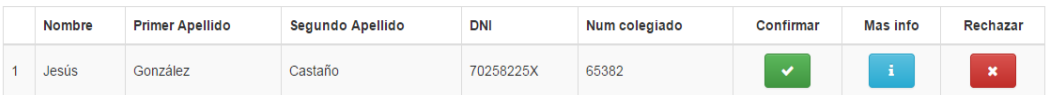

Figura 6.1.5. Tabla de peticiones de alta a la aplicación.

La opción de "Confirmar", ejecutará la lógica necesaria para que el médico quede registrado en la base de datos, en la colección "usuarios". Además, quedará eliminado al instante de la colección "peticionAlta" y, por tanto, de esta lista de peticiones. La opción de "Más info", permitirá al administrador comprobar los datos insertados por el médico en el proceso de registro. La opción de "Rechazar", anulará el proceso de registro en la aplicación.

Implementación 3. Dar de baja a un médico.

En la implementación de esta funcionalidad, el médico proporcionará al administrador sus datos personales para darse de baja de la aplicación. Es entonces cuando el administrador insertará el DNI del médico en el formulario de baja y se procederá a ejecutar la lógica del mismo (Figura 6.1.6).

Primero se hará una búsqueda sobre la colección de “usuarios” para encontrar al médico concreto, y después se procederá a ejecutar la instrucción “remove” sobre ese DNI, eliminando así el documento del médico de la colección “usuarios”, evitando poder hacer un borrado múltiple. Con ello se asegura mantener la coherencia y consistencia de la base de datos en cuanto a altas/bajas, para evitar duplicados o informaciones inconsistentes.

Figura 6.1.6. Formulario de baja de médico.

Implementación 4. Buscar y actualizar un médico.

Si se produce algún cambio relevante en los datos de un médico, el administrador tiene la opción de buscar su perfil y modificar ciertos datos, gracias a su formulario correspondiente (Figuras 6.1.7 y 6.1.8). Se insertará el DNI del médico para acceder mediante POST a sus datos, que se cargarán a su vez en otro formulario, pudiendo así enviar estas modificaciones directamente a la base de datos, sobrescribiendo los campos que se hayan cambiado y manteniendo aquellos que no.

Figura 6.1.7. Formulario de consulta de médico.

Figura 6.1.8. Formulario de modificación de médico.

En cuanto a lógica simplemente se busca al médico mediante un findOne, como se ha estado haciendo en todos los casos. Así se evita acceder a perfiles que no sean los que interesan, y se lanza un update sobre el documento concreto que se quiere modificar, cogiendo los datos del formulario y metiéndolos en un array de nuevos datos, que será el que actualice el perfil, gracias al comando set (Figura 6.1.9).

```
$doc = array(
    "DNI"=>$dni,
    "Número_de_colegiado"=>$numCol,
    "Apellido_1"=>$apellido1,
    "Apellido_2"=>$apellido2,
    "Nombre"=>$nombre,
    "Teléfono"=>$telefono,
    "Dirección"=>$direccion,
    "Especialidad_Médica"=>$espemedica,
    "Hospital"=>$hospital
);

$nuevosdatos = array('$set'=>$doc);
$coll->update(array("DNI"=>$dni),$nuevosdatos);
```

Figura 6.1.9. Código modificación médico.

Las funcionalidades implementadas para el médico son las siguientes:

Implementación 1. Dar de alta paciente.

Este proceso funcionará de forma análoga a como lo hace el alta de médicos por parte del administrador. Tendrá que acceder a su formulario correspondiente de altas, introducir los datos personales del paciente, y mediante una llamada POST, se insertará un documento en la tabla de “usuarios”, teniendo en cuenta que se hará con el DNI, tanto de médico, como de paciente, relacionándolos directamente, por lo que ningún otro médico tendrá acceso al historial de dicho paciente, además de crear una carpeta en el sistema de ficheros (Figura 6.1.10). Dicha carpeta que tendrá cada paciente nada más se inserte en el sistema, será unívoca ya que se le asignará el DNI del paciente concreto, y es ahí donde se almacenarán los documentos asociados.

Figura 6.1.10. Formulario de alta de pacientes.

Cabe destacar que en caso de que el paciente exista, no se podrá realizar la inserción. Para comprobarlo se realiza un findOne en la colección "pacientes" y si no se encuentra se habilita la inserción. En caso contrario se cancela el proceso y se avisa del error (Figura 6.1.11).

```
foreach($_POST as $nombre_campo => $valor){
    $doc[$nombre_campo] = $valor;
}
$pac->insert($doc);
```

Figura 6.1.11. Código de alta de pacientes.

Implementación 2. Dar de baja pacientes.

De nuevo, un proceso parecido al alta de pacientes. En este caso se tendrá también coherencia con la base de datos, como es lógico, ya que cada médico sólo podrá eliminar pacientes asociados a sí mismo. Cuando se quiere eliminar a un paciente, lo que se hace es buscarle en la lista de pacientes y elegir la opción de eliminar (Figura 6.1.12).

Figura 6.1.12. Tabla de baja de pacientes.

La aplicación obtendrá los datos de la tabla y hará una búsqueda del documento en la base de datos. Acto seguido lo eliminará gracias a la instrucción remove (Figura 6.1.13).

```php
$res = $pac->findOne(array('DNI'=>$dniPaciente),array('_id' => 0));
$otro = $pac->remove(array('DNI'=>$dniPaciente));


//eliminacion de documentos asociandos
if (file_exists("..//documentos//" . $dniPaciente . "//" )) {
    eliminarDir("..//documentos//" . $dniPaciente . "//" );
}

//elimiancion de test asociados
$unomas = $db->tests;
$cursor = $unomas->find();
$cursor = $unomas->remove(array('DNIPaciente'=>$dniPaciente), array('DNIMedico'=>$dni));

function eliminarDir($carpeta)
{
    foreach(glob($carpeta . "/*") as $archivos_carpeta)
    {
        if (is_dir($archivos_carpeta))
            eliminarDir($archivos_carpeta);
        else
            unlink($archivos_carpeta);
    }
    rmdir($carpeta);
}
```

Figura 6.1.13. Código de baja de pacientes.

Implementación 3. Buscar y actualizar un paciente.

En este caso hay dos opciones por las que se puede optar según la lógica de la aplicación. Por un lado, la opción de consultar los datos de un paciente (Figura 6.1.14). Por otro, lado, la posibilidad de modificarlos (Figura 6.1.15).

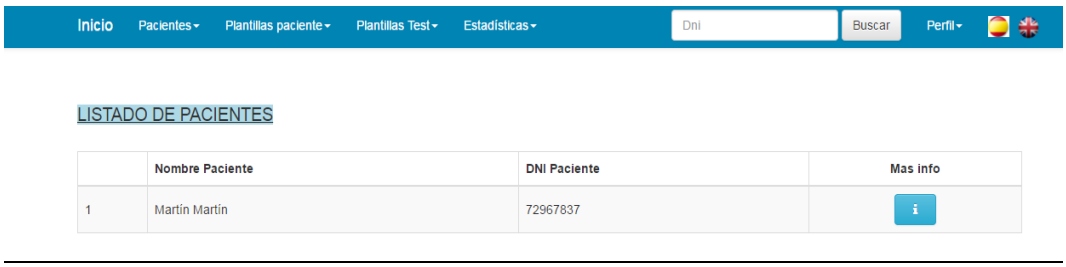

Figura 6.1.14. Tabla de consulta de pacientes.

Figura 6.1.15. Tabla de modificación de pacientes.

La primera lo que hace es acceder al documento asociado en la colección “usuarios” de la base de datos e ir cargando, de forma tabulada, los datos personales, por un lado, y sus documentos asociados, por otro.

Pudiendo así acceder, tanto a las funcionalidades de exportar datos personales a pdf, como exportar documentos asociados a zip, y también poder consultar los test que se hayan realizado al paciente en cuestión (Figuras 6.1.16 y 6.1.17).

```php
foreach ($cursor as $documento) {
    if($documento['DNIPaciente'] == $dniPaciente  and $documento["DNIMedico"] == $dni ) {
        $name = $documento["Nombre"];
        echo "<li> <a href='muestraTest.php?DNIPaciente=$dniPaciente&Nombre=$name' >" . $documento["Nombre"] . "</a>";
    }
}
```

Figura 6.1.16. Código de muestra de links a tests de un paciente.

```php
if (file_exists("..//documentos//" . $dniPaciente . "//" )) {
    $directorio = opendir("..//documentos//" . $dniPaciente . "//" ); //ruta actual
    while ($archivo = readdir($directorio)) //obtenemos un archivo y luego otro sucesivamente
    {
        if (is_dir($archivo))//verificamos si es o no un directorio
        {
            //de ser un directorio lo envolvemos entre corchetes
        }
        else
        {
            echo "<li> <a href=' ..//documentos//$dniPaciente//$archivo' target='_blank'>" . $archivo . "</a>";
        }
    }
}
```

Figura 6.1.17. Código de muestra de documentos asociados de un paciente.

La segunda lo que se encarga es de hacer esto mismo, pero con la opción de poder modificar los campos cargados en la tabla, de tal manera que, al elegir la opción de modificar, se hará un update en el documento con ese DNI guardado en la colección de usuarios. Para que la modificación se realice correctamente se tienen que guardar los datos nuevos, extraídos de los campos del formulario en un nuevo array, y modificar el documento de la base de datos haciendo un set con este nuevo array dentro de la instrucción de update.

Implementación 4. Exportar a pdf.

Esta implementación es accesible desde la pantalla de consulta de un paciente. Una vez en esta página, se podrá exportar la tabla de datos personales simplemente pulsando en el icono de PDF (Figura 6.1.18). Esto lo que desencadenará es la llamada a través de un enlace (oculto en el icono antes mencionado) a una página que desarrollará la lógica. En este proyecto se ha utilizado TCPdf para realizar esta implementación, ya que lo que hace es transformar el html que nos interesa y plasmarlo en un pdf de forma sencilla.

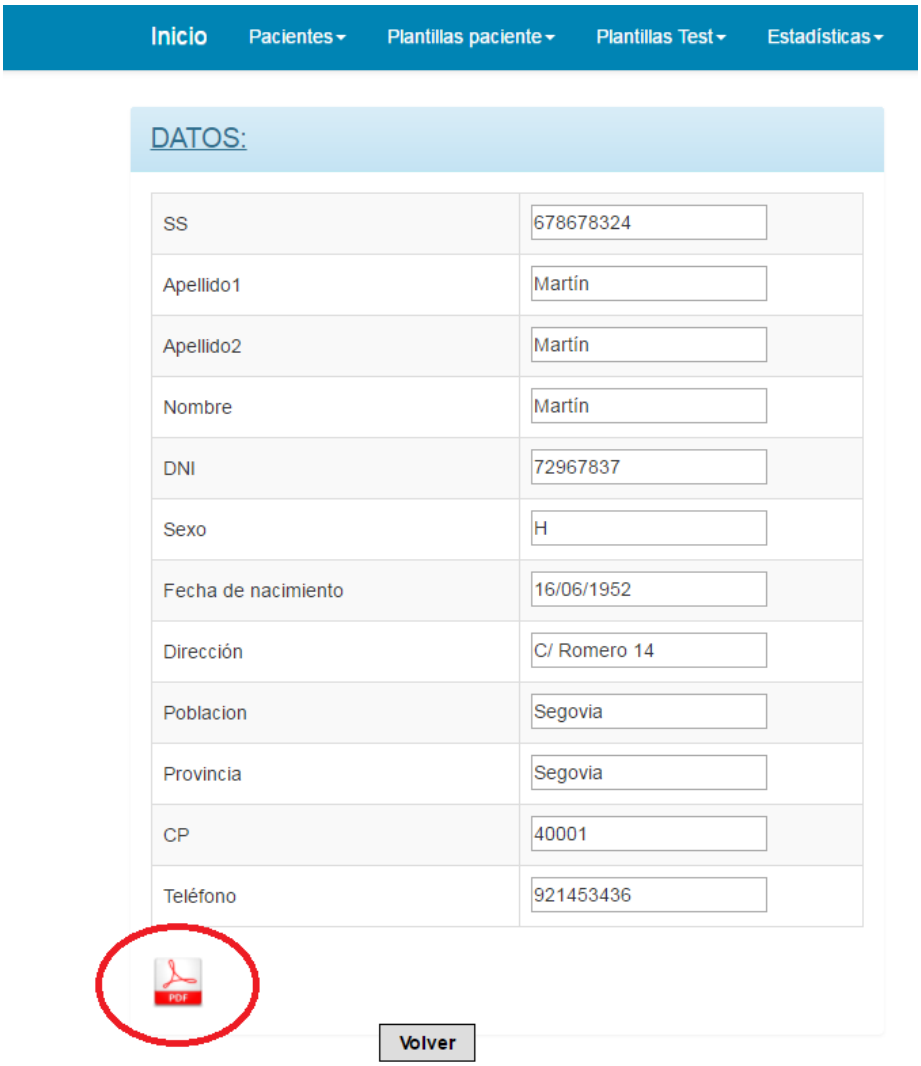

Figura 6.1.18. Exportar a pdf

Una vez se genera dicho pdf se muestra en una pantalla externa mediante la adición de _blank al enlace, de tal manera que no interfiere en el flujo de la aplicación, y se le da la opción al médico de guardarlo donde él quiera, en su local.

Implementación 5. Subir documentos adjuntos a un paciente.

En este caso, se subirá un documento pdf asociándolo al paciente (Figura 6.1.19) que se copiará directamente en la carpeta con el DNI del paciente concreto (utilizando la función getcwd), creada previamente, al dar de alta al paciente. Esto se hace así porque en caso de alguna anomalía en la creación del directorio del paciente, esta función devolverá false, con lo que se cancelará el proceso. Si el proceso se completa satisfactoriamente, aparecerá en el bloque de documentos asociados de forma visual en la aplicación web.

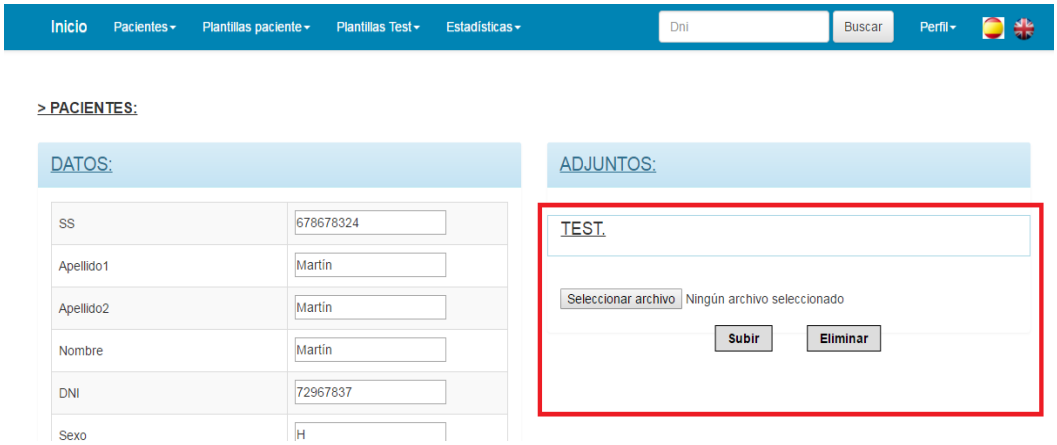

Figura 6.1.19. Subir documentos

Implementación 6. Exportar documentos adjuntos a zip.

En esta funcionalidad, se da la opción de exportar la lista de documentos asociados del paciente, en un único fichero zip (Figura 6.1.20). Para ello, primero se tendrá que comprobar, que efectivamente, el paciente existe en la base de datos. Se comprueba aquí que el DNI del paciente existe como nombre de fichero en nuestro sistema de ficheros. Esta carpeta, como se explicó en la implementación del alta de pacientes, se crea en el momento de entrada en el sistema del paciente en cuestión.

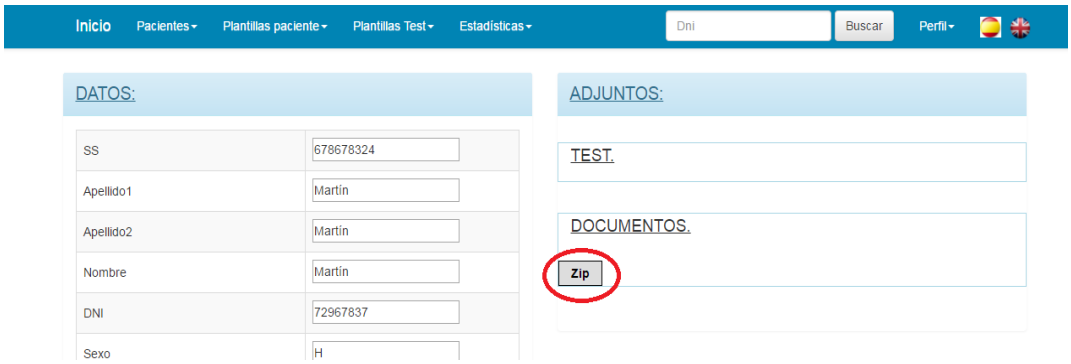

Figura 6.1.20. Exportar documentos

Para mantener una lógica común y guardar coherencia, el zip recibirá por nombre, el propio DNI del paciente. Por tanto, cuando se elige la opción de exportar fichero zip, lo que ocurre es que en el directorio con DNI “n” se creará un zip “n.zip”, que contendrá todos sus documentos asociados. La forma de obtener dichos documentos es recorriendo el directorio, documento a documento y añadiéndolo al zip, uno por uno, mediante la instrucción addFile, donde se le insertan dos parámetros, la ruta donde se almacenará (en nuestro caso el zip dentro de la carpeta personal del paciente) y el fichero.

Implementación 7. Crear plantillas de registro.

Esta implementación consta de dos partes muy diferenciadas. Una, la generación dinámica de campos con jQuery[6] en el formulario de creación (Figura 6.1.21), y otra, el guardado en la base de datos de dicha plantilla.

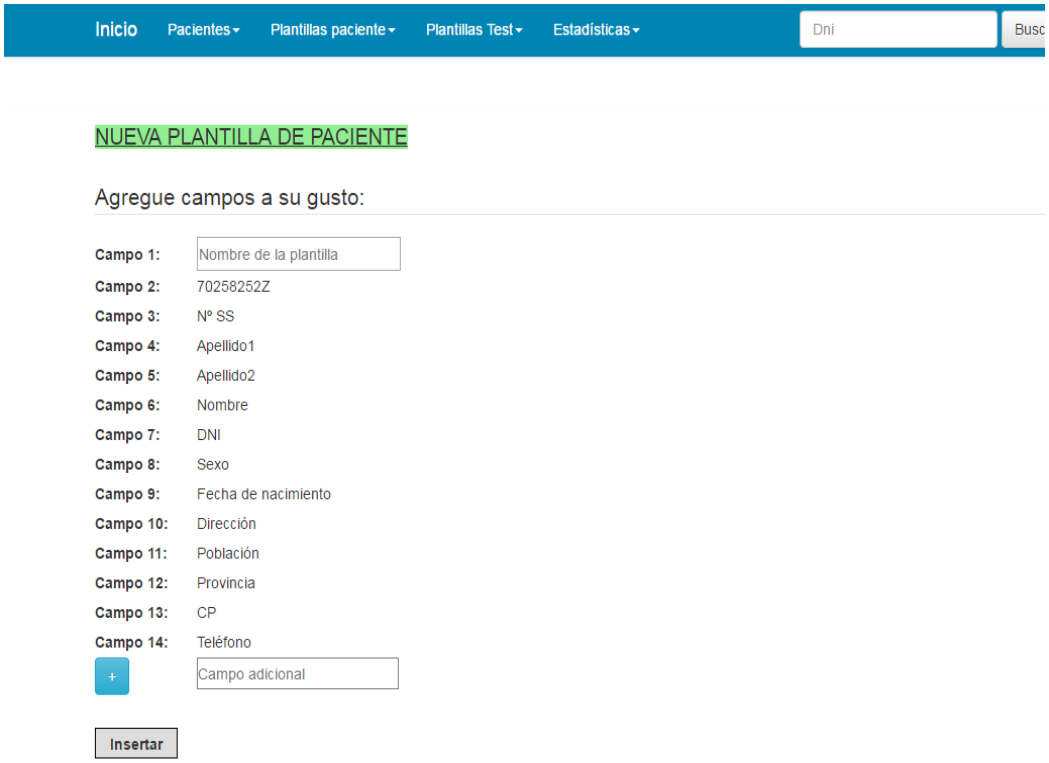

Figura 6.1.21. Formulario de creación de plantillas de registro de pacientes.

Para la primera parte, se dispondrá de tres variables, que nos ayudarán a generar los campos extra de forma dinámica.

Maxinputs indica el número máximo de campos posibles a generar, en este caso 99 era una opción más que razonable. La variable contenedora nos ayuda a generar una id diferente para cada nuevo campo, lo que es necesario si se quiere que cada campo se guarde correctamente en la base de datos. Y, por último, AddButton, que sirve para poder pintar el botón que irá mostrando un campo nuevo cada vez.

Además de esto, al tener siempre un número fijo de campos obligatorios, se tenía que empezar, por estructura de base de datos, a generar las id de los campos nuevos a partir de este número fijo (Figura 6.1.22).

```
$(document).ready(function() {

    var MaxInputs       = 99; //Número Maximo de Campos
    var contenedor      = $("#contenedor"); //ID del contenedor
    var AddButton       = $("#agregarCampo"); //ID del Botón Agregar

    //var x = número de campos existentes en el contenedor
    var x = $("#contenedor div").length + 1;
    var FieldCount = x+13; //para el seguimiento de los campos

    $(AddButton).click(function (e) {
        if(x <= MaxInputs) //max input box allowed
        {
            FieldCount++;
            //agregar campo
            $(contenedor).append('<div><input type="text" class="campoextra" name="Campo '+ FieldCount +'" id="campo_'+ FieldCount +'" placeholder="Campo adicional"/></div>');
            x++; //text box increment
        }

        return false;
    });

});
```

Figura 6.1.22. Código generación campos dinámicos.

La segunda parte, recibe a través de ese formulario ya generado, y mediante POST, todos los campos completos. Es aquí cuando a través de un insert en base de datos se genera la plantilla adjunta a ese médico. Y es solo suya. Ningún otro médico puede utilizarla en un futuro.

Pero para llegar a esta inserción, y que todo se comporte de forma ordenada, primero se ha tenido que construir el sistema. En el paso donde se generan los campos extra, se tiene como clave (los labels del formulario) una lista de valores (Campo 1, …, Campo n). Pues bien, estas claves, contienen en su par valor, el nombre que el campo tomará de cara al usuario, en este caso el médico. Y, por consiguiente, se insertarán en este orden en la base de datos (Figura 6.1.23).

```
$doc = array();
$t = 1;
$parse =  str_replace(' ','_',$_POST['Nombre_de_la_plantilla']);
$funciona = array("Campo_1"=>$parse, "Campo_2"=>$_POST['Médico']);
echo $funciona;
$res = $plan->remove($funciona);

foreach($_POST as $nombre_campo => $valor){
   $campo = "Campo_" . $t;
   IF(is_null($valor) or $valor == "")
   {
   }
    else {
      $valor = str_replace(' ','_',ucfirst($valor));
      $doc[$campo] = $valor;
      $t++;
      }
}
```

Figura 6.1.23. Código traspaso clave-valor.

El siguiente paso de la lógica es transformar esto y hacer posible que estos valores sean acompañados de un input donde introducir los datos. Para ello, se carga un nuevo formulario, donde el valor de la inserción anterior, se generará en el label del formulario, convirtiéndose así en clave, y dejando el valor vacío para ser rellenado. Y el último paso es volver a coger estos valores y procesarlos de la manera habitual.

Implementación 8. Crear plantilla de test.

Esta implementación funciona en cuanto a lógica igual que la explicada anteriormente para el caso de plantillas de paciente, con la única diferencia de que, al tener sólo tres campos fijos, la id de los elementos se genera a partir de este número, y se ha adaptado para este caso concreto (Figura 6.1.24). Aun así, toda la lógica y construcción de los campos dinámicos y su inserción, en este caso en su colección propia (no comparte con la de generación de plantillas de pacientes), funciona de forma idéntica.

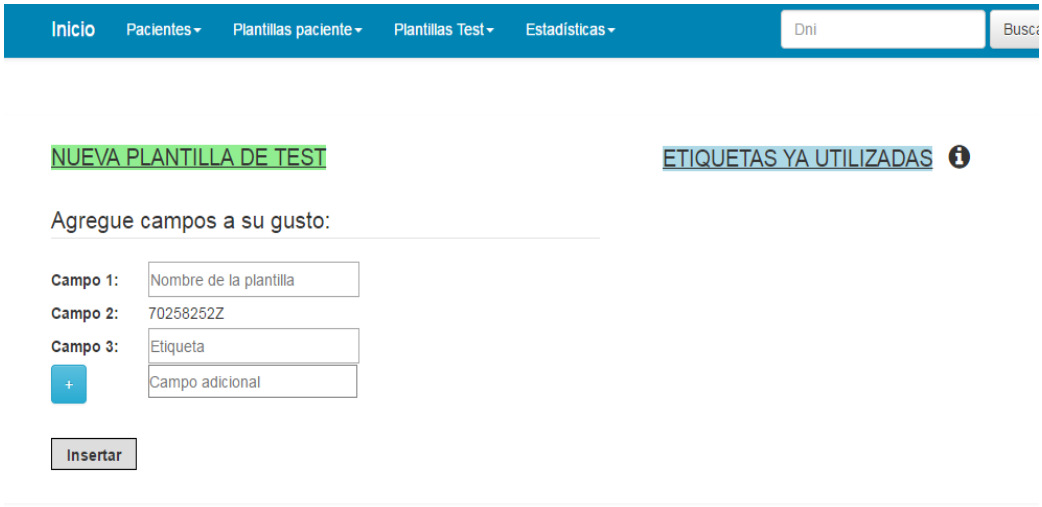

Figura 6.1.24. Formulario de creación de plantillas de test.

Implementación 9. Generar estadísticas.

Para poder explotar la información de la base de datos, se ha generado una implementación sobre una serie de estadísticas a elegir, en función a tres valores.

Estos tres valores son, la enfermedad a buscar, rango de edades y sexo de los pacientes. Todas ellas se pueden combinar entre sí, pudiendo buscar estadísticas con solo un campo, dos campos, o todos a la vez, en cuyos casos, se interrelacionan de maneras diferentes (Figura 6.1.25).

Figura 6.1.25. Formulario de elección de valores para generar estadísticas.

Por ejemplo, si se rellena el campo de sexo y el rango de edades, se mostrará una gráfica de tipo tarta, mostrando las enfermedades que han cumplido los dos requisitos anteriores.

Para que esto funcione, se ha utilizado un módulo llamado Chart.js, que se encarga de generar las gráficas a través de un elemento html llamado canvas. Dicha gráfica tiene una serie de componentes internos para su construcción, como datasets, para cargar los valores que se quieren pintar en ella, elementos css, labels para indicar que es cada parte de la tarta etc. Los colores de la misma se generan aleatoriamente gracias a una función randomize, que genera el color en hexadecimal.

La lógica funciona con una serie de ifs, que irán eligiendo el caso concreto de búsqueda de estadísticas en función de los valores recibidos del formulario anterior. Se harán las búsquedas pertinentes sobre los documentos que se ajusten a los criterios de búsqueda y se guardarán los valores en un array, para poder pasárselos al elemento canvas, y que el javascript de la librería Chart.js lo interprete. De este modo se pintará la gráfica con la búsqueda realizada.

Por último, se concluye esta sección con sendos diagramas, que explican la interrelación entre cada script php utilizado en el proyecto, separados por funcionalidades. El primer diagrama representa los scripts utilizados para las operaciones CRUD (Figura 6.1.26), el segundo diagrama representa aquellos scripts que sirven para la creación de plantillas (Figura 6.1.27) y, por último, un diagrama con otras funcionalidades importantes (Figura 6.1.28).

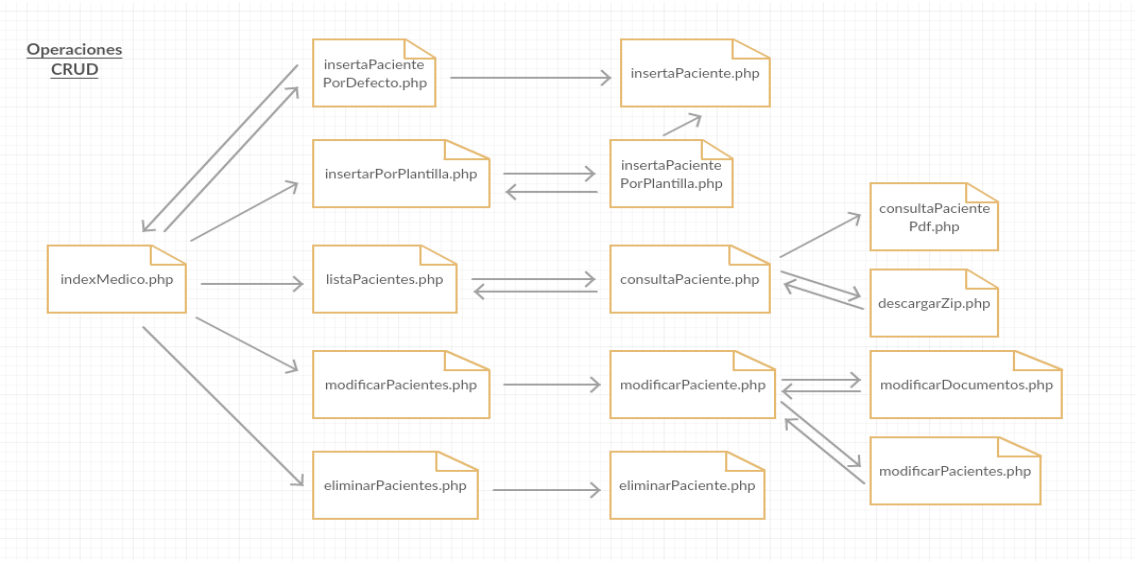

Figura 6.1.26. Diagrama de scripts. Operaciones CRUD

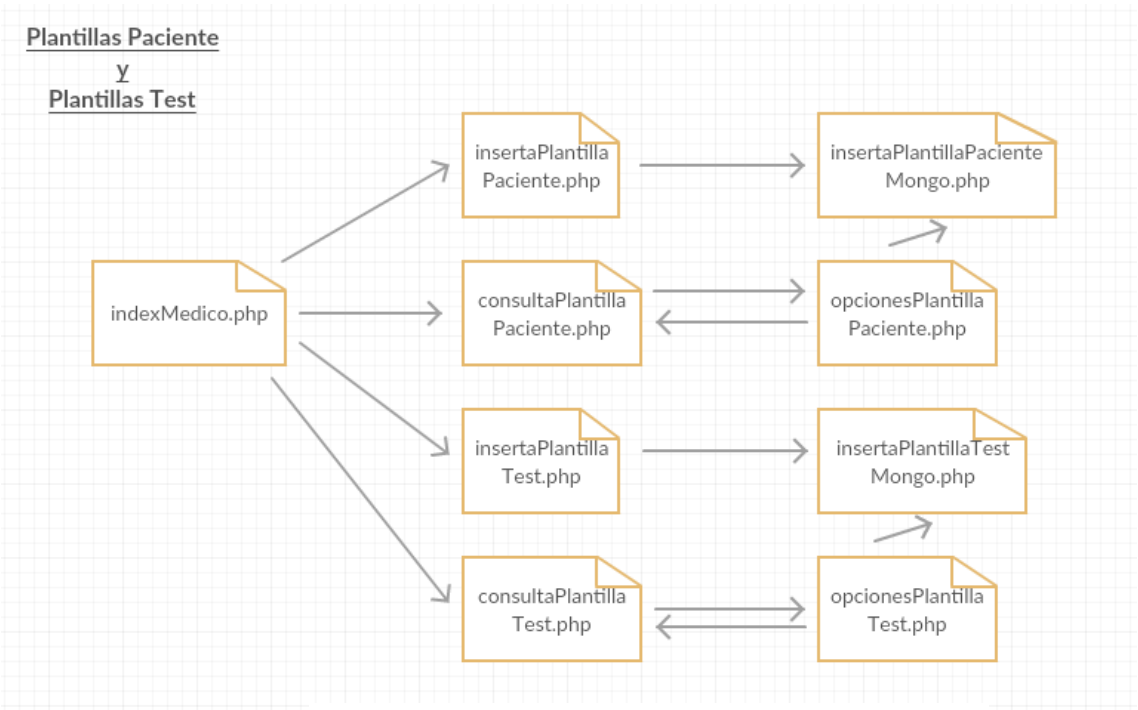

Figura 6.1.27. Diagrama de scripts. Plantillas paciente y plantillas test

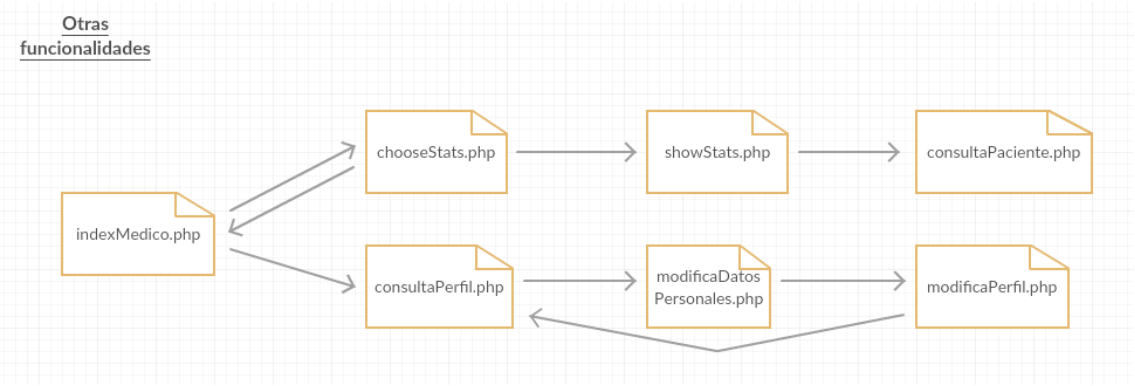

Figura 6.1.28. Diagrama de scripts. Otras funcionalidades

6.2 Aplicación móvil

La implementación de la aplicación Android se ha realizado mediante el programa Android Studio y el lenguaje de programación Java.

El proyecto consta de dos apartados fundamentales, la carpeta “libs” y la carpeta “src” (Figuras 6.2.1 y 6.2.2). La carpeta “libs” contiene las librerías fundamentales para la implementación del proyecto. Cabe destacar sobre las demás la librería mongo-java-driver-3.4.0-SNAPSHOT.jar dedicada a la conexión con MongoDB.

Dentro de la carpeta “src” se encuentra todo el código y la implementación de la aplicación dividiéndola en tres subapartados, las carpetas “java”, “res” y el documento “AndroidManifest.xml”.

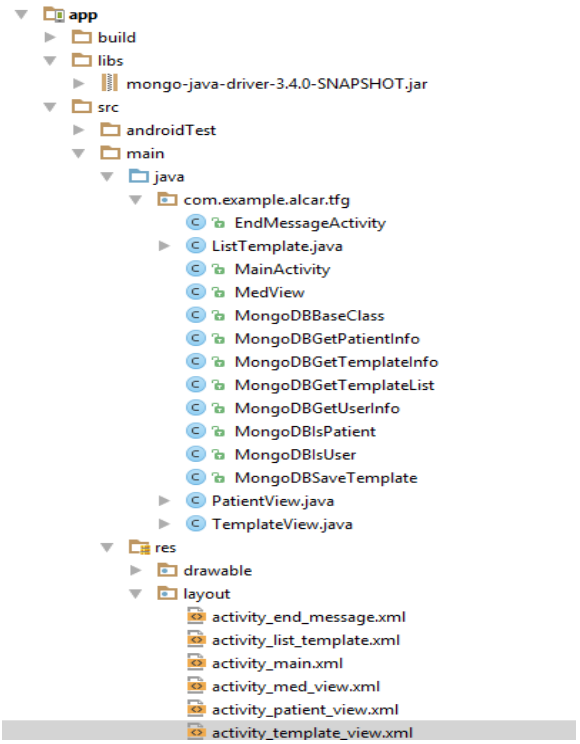

Figura 6.2.1. Estructura de proyecto Android 1.

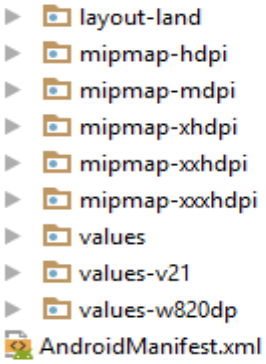

Figura 6.2.2 Estructura de proyecto Android 2.

El documento, AndroidManifest.xml contiene las líneas imprescindibles para permitir a la aplicación Android conectarse a Internet e interactuar con la base de datos de MongoDB la cual a su vez se encuentra on-line esperando consultas (Figura 6.2.3).

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Figura 6.2.3. Código Android Studio para la conexión a internet

En la carpeta “java” se encuentran todos los documentos de formato Java sobre los que va a correr la aplicación. Se dividen en dos apartados diferenciados, aquellos relacionados con la conexión con la base de datos, cuyo nombre comienza con “MongoDB...,” y aquellos destinados al control de las distintas pantallas e interfaces de las que se compone la aplicación y son aquellos terminados en “...Activity” ó “...View”.

Estos documentos “...Activity” permite el paso entre ellos con la generación de objetos Intent. Un objeto Intent es una descripción abstracta de una operación a realizar, suele crearse a la hora de pulsar un botón y permiten además el paso de variables entre vistas, lo que es imprescindible a la hora de mantener siempre la sesión del médico abierta mientras esté trabajando.

La carpeta “res”, incluye varias subcarpetas con información del proyecto, pero a la hora de la implementación la importante es “layout”. Esta carpeta contiene todos los ficheros .xml destinados a la construcción de la interfaz de la aplicación, la situación de textos, botones, listas, etc.

En la aplicación móvil únicamente va a acceder el médico, puesto que será el encargado de la realización de test a sus pacientes, con lo que las siguientes implementaciones irán dirigidas a dicho médico.

Implementación 1. Búsqueda y acceso al historial de un paciente.

Esta opción permitirá al médico visualizar la información relativa a un paciente en la aplicación móvil.

Para ello, el primer paso que tiene que hacer el médico en la aplicación es identificarse con su usuario y contraseña en la pantalla correspondiente. Al pulsar sobre el botón “Acceder”, el cual solo se permitirá pulsar cuando ambos campos están rellenos, obtendrá mediante un objeto Listener los datos de los campos, y llamará a la clase que comprueba el usuario, “MongoDBIsUser” (Figura 6.2.4 y 6.2.5).

```
@Override
public Boolean call() throws Exception {
        try {
                MongoClient mongoClient = new MongoClient(getIp(), getPort());
                MongoDatabase prueba = mongoClient.getDatabase("prueba");
                MongoCollection<Document> usuario = prueba.getCollection("usuarios");
                Document dni = usuario.find(eq("DNI", this.user)).first();
                if(dni != null) {
                        Object contraseña1 = dni.get("Contraseña");
                        if(contraseña1.equals(this.pass))
                        {
                                mongoClient.close();
                                return true;
                        }
                        else{
                                mongoClient.close();
                                return false;
                        }
                }
                else
                {
                        mongoClient.close();
                        return false;
                }

        } catch (Exception e) {
                System.out.println(e.toString());
                return false;
        }
}
```

Figura 6.2.4. Código Android Studio para control de acceso de médicos

```
@Override
public Boolean call() throws Exception {
    try {
        MongoClient mongoClient = new MongoClient(getIp(), getPort());
        MongoDatabase prueba = mongoClient.getDatabase("prueba");
        MongoCollection<Document> usuario = prueba.getCollection("pacientes");

        BasicDBObject andQuery = new BasicDBObject();
        List<BasicDBObject> obj = new ArrayList<~>();
        obj.add(new BasicDBObject("Médico", user));
        obj.add(new BasicDBObject("DNI", patient));
        andQuery.put("$and", obj);
        Document dni = usuario.find(andQuery).first();

        if(dni != null ) {
                mongoClient.close();
                return true;
        }
        else
        {
            mongoClient.close();
            return false;
        }

    } catch (Exception e) {
        System.out.println(e.toString());
        return false;
    }
}
```

Figura 6.2.5. Código Android Studio para control de existencia de pacientes

Esta clase hace una búsqueda en la base de datos el DNI introducido por el médico, si existe obtiene el campo contraseña y lo equipara al introducido por el médico. En caso de no coincidir se mostrará un mensaje de alerta. Si se introducen los campos correctamente se pasará a la vista de búsqueda de paciente, “activity_med_view.xml” donde se saluda al médico con un mensaje de bienvenida y se le ofrece buscar al paciente (Figura 6.2.6).

Figura 6.2.6. Captura de implementación de la pantalla de búsqueda de pacientes

Al pulsar el botón de “Buscar Paciente” se llama a la clase “MongoDBIsUser.jar”, que comprobará en la base de datos si el DNI introducido en el campo coincide con algún paciente de ese médico. En el caso en el que exista dicho paciente, se creará un nuevo objeto Intent como se explicó previamente, que realizará el cambio de pantalla. Es en esta vista donde se encontrará la información relativa al paciente, nombre, apellidos, sexo, fecha de nacimiento, etc (Figura 6.2.7).

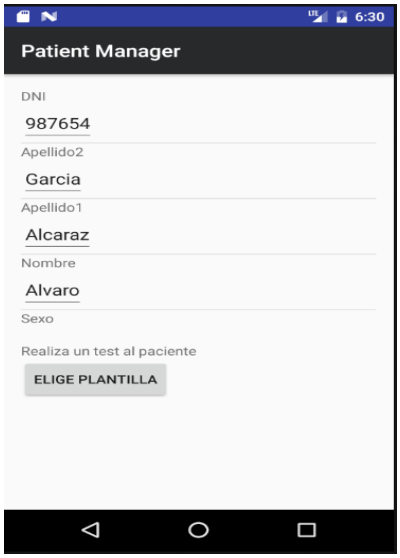

Figura 6.2.7. Captura de implementación de la pantalla de información de pacientes

Implementación 2. Agregar un test a un paciente.

Esta es la principal función de la aplicación móvil y el motivo de su diseño. De nuevo al igual que en el caso anterior el médico debe acceder con sus credenciales, así como buscar un paciente que exista y al que esté tratando.

Una vez el médico haya introducido un DNI de paciente correcto y se encuentre en la pantalla de la información aparecerá en la parte inferior el botón para realizar el test. De

nuevo mediante un objeto Intent (Figura 6.2.8). En este caso se pasará a la clase ListTemplate.jar, esta clase se rellena mediante el acceso a la base de datos Mongo y la recuperación de todas aquellas plantillas que haya creado el médico logueado. Produciéndose la particularidad de no haber creado ninguna plantilla de test le saldrá un mensaje informativo haciéndolo saber.

```
public class ListTemplate extends AppCompatActivity {

    String user = null;
    String patient = null;
    ListView lista = null;
    ArrayList<String> result = new ArrayList<>();
    TextView textView2 = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_template);

        lista = (ListView) findViewById(R.id.list);

        Intent intent = this.getIntent();
        user = intent.getStringExtra("usuario");
        patient = intent.getStringExtra("paciente");

        //Creamos la task para el acceso a la informacion del medico, la guardaremos en un Hash Map
        FutureTask task = new FutureTask(new MongoDBGetTemplateList(user, patient));
        ExecutorService es = Executors.newSingleThreadExecutor();
        es.submit(task);
        try {

            result = (ArrayList<String>) task.get();


        } catch (Exception e) {
            System.err.println(e);
        }
        //Creamos el adaptador de la lista
        AdaptadorListaTemplate adaptadorLista = new AdaptadorListaTemplate();

        String[] strong = {"Uno", "Dos", "Cuatro"};
        if (result != null) {
            for (int i = 0; i < result.size(); i++) {
                adaptadorLista.cosas.add(new OpcionEncuestaTextoTemplate(result.get(i).toString(), ListTemplate.this, this.user,this.patient));
            }
        }
        else{
            textView2 = (TextView) findViewById(R.id.wrong);
            textView2.setText("No tiene creada ninguna plantilla de test. Por favor cree una en la aplicación web.");
        }
        lista.setAdapter(adaptadorLista);
    }
```

Figura 6.2.8. Código Android Studio para mostrar la lista de plantillas de test

Figura 6.2.9. Pantalla para mostrar la lista de plantillas de test

Tras haber introducido un DNI correcto, y tener creada al menos una plantilla de test aparecerá una lista con los nombres de todas las plantillas acompañados de un botón "Elegir" para que el médico pueda elegir la que mejor le convenga en cada situación (Figura 6.2.9). De nuevo se utiliza un objeto Intent para el paso de pantallas, en este caso la clase java que implementa el activity "activity_template_view.xml" será TemplateView.java.

```
@Override
View getVista() {
    LinearLayout l = null;
    Button boton = null;
    l = new LinearLayout(this.ctx);
    l.setOrientation(LinearLayout.HORIZONTAL);
    textView = new TextView(this.ctx);
    textView.setText(this.titulo.replace("_"," "));
    LinearLayout k = null;
    k = new LinearLayout(this.ctx);
    //k.layout(0,0,0,0);
    k.setLayoutParams(new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, 200,200));
    k.setGravity(Gravity.RIGHT);
    boton = new Button(this.ctx);
    boton.setText("Elegir");
    boton.setOnClickListener((v) -> {
            Intent intent = new Intent((v.getContext()), TemplateView.class);
            intent.putExtra("usuario",usuario);
            intent.putExtra("paciente", patient);
            intent.putExtra("name", titulo);
            v.getContext().startActivity(intent);

    });

    l.addView(textView);
    k.addView(boton);
    l.addView(k);
    return l;
```

Figura 6.2.10. Código Android Studio para el cambio entre pantallas.

Dicha clase se alimentará de la búsqueda de la obtención del objeto mongo correspondiente al test seleccionado anteriormente. Para poder realizar la muestra de todos los campos de los que consta el test se recurre a la creación de un objeto que extienda la clase BaseAdapter (Figura 6.2.11).

```
class AdaptadorListaTemplateInfo extends BaseAdapter {

    public ArrayList<OpcionEncuestaTemplateInfo> cosas = new ArrayList<~>();

    @Override
    public int getCount() { return cosas.size(); }

    @Override
    public Object getItem(int position) { return cosas.get(position); }

    @Override
    public long getItemId(int position) { return position; }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        return cosas.get(position).getVista();
    }
}

abstract class OpcionEncuestaTemplateInfo {
    abstract View getVista();
    abstract String getTitulo();
    abstract String getValor();
}
```

Figura 6.2.11. Código Android Studio para extender la clase BaseAdapter.

BaseAdapter es una clase común de Android Java destinada a la creación de distintos tipos de vista personalizada. Permite crear clases que la extienda de modo que luego objetos de estas clases puedan ser añadidos a una lista y mostrados en tiempo de ejecución en la pantalla.

El objeto BaseAdapter permite crear en este caso, por cada campo dentro de la plantilla de test, una línea con el campo a preguntar y un campo de edición de texto para que el médico escriba la respuesta del paciente (Figura 6.2.14).

En posibles evoluciones de la aplicación se podría ofrecer la posibilidad de introducir distintos tipos de campos que no sean solo texto plano como se ofrece actualmente, cabe recordar que esto solo es un prototipo inicial.

Una vez el médico considere que ha finalizado el test dispondrá de un botón en la parte inferior de la lista, el botón "Finalizar", al pulsarlo la aplicación recorrerá todos los campos de la lista creada, tomando el título y el valor rellenado por el médico y guardándolos en su correspondiente tabla de la base de datos (Figuras 6.2.12 y 6.2.13).

```
public void clickSave (View view) {

    HashMap<String, String> hasmapo = null;
    hasmapo = new HashMap<String,String>();
    hasmapo.put("Etiqueta",etiqueta);
    for (int i = 0; i < adaptadorLista.getCount();  i++) {
        String titulo = adaptadorLista.cosas.get(i).getTitulo();
        String valor = adaptadorLista.cosas.get(i).getValor();
        hasmapo.put(titulo,valor);
    }
    FutureTask task = new FutureTask(new MongoDBSaveTemplate(user,patient, hasmapo));
    ExecutorService es = Executors.newSingleThreadExecutor();
    es.submit(task);
    boolean result = false;
    try {
        result = (boolean) task.get();
    } catch (Exception e) {
        System.err.println(e);
    }
    if (result) {
        Intent intent = new Intent(this, EndMessageActivity.class);
        intent.putExtra("usuario", user);
        startActivity(intent);
    }
}
```

Figura 6.2.12. Código Android Studio para la función de llamada al guardado.

```
@Override
public Boolean call() throws Exception {
    try {

        MongoClient mongoClient = new MongoClient(getIp(), getPort());
        MongoDatabase prueba = mongoClient.getDatabase("prueba");

        MongoCollection<Document> tests = prueba.getCollection("tests");

        org.bson.Document document1 = new org.bson.Document();
        document1.put("DNIPaciente",patient);
        document1.put("DNIMedico",user);
        Calendar c = Calendar.getInstance();

        SimpleDateFormat df = new SimpleDateFormat("dd-MMM-yyyy");
        String formattedDate = df.format(c.getTime());
        SimpleDateFormat df2 = new SimpleDateFormat("HH:mm:ss");
        String formattedDate2 = df2.format(c.getTime());

        name = patient + "-" + formattedDate + "-" + formattedDate2;
        document1.put("Nombre", name);
        for(Map.Entry<String, String> entry : hasmapo.entrySet()) {
            String key = entry.getKey();
            String value = entry.getValue();
            document1.put(key,value);
        }
        tests.insertOne(document1);
        mongoClient.close();
        return true;

    } catch (Exception e) {
        System.out.println(e.toString());
        return null;
    }
}
```

Figura 6.2.13. Código Android Studio para la función que realiza el guardado.

Figura 6.2.14. Pantalla para la muestra de las preguntas del test

Por último, cierran esta sección dos diagramas de las clases utilizadas en la implementación de la aplicación móvil.

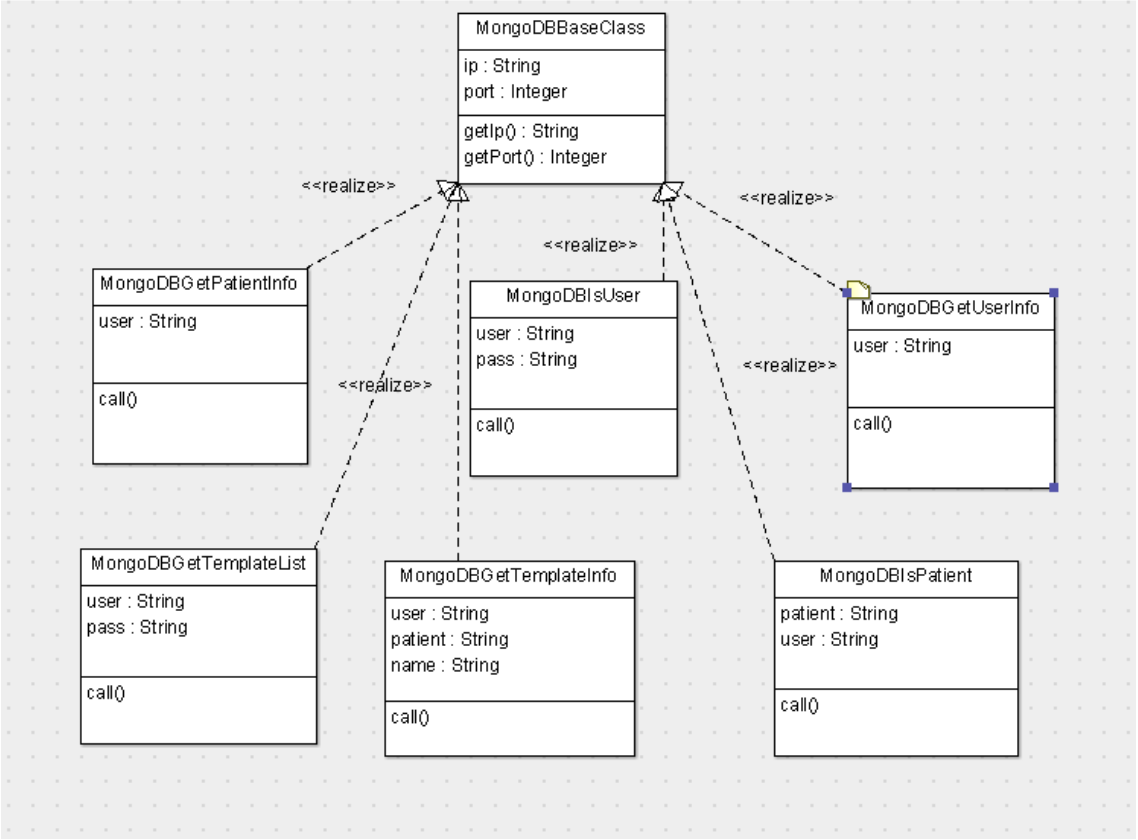

Figura 6.2.15. Diagrama de clases app móvil.

Todas las clases de control de base datos extienden a una superclase MongoDBMainClass creada para controlar la conexión, donde se introduce la IP y el puerto común a todas (Figura 6.2.15).

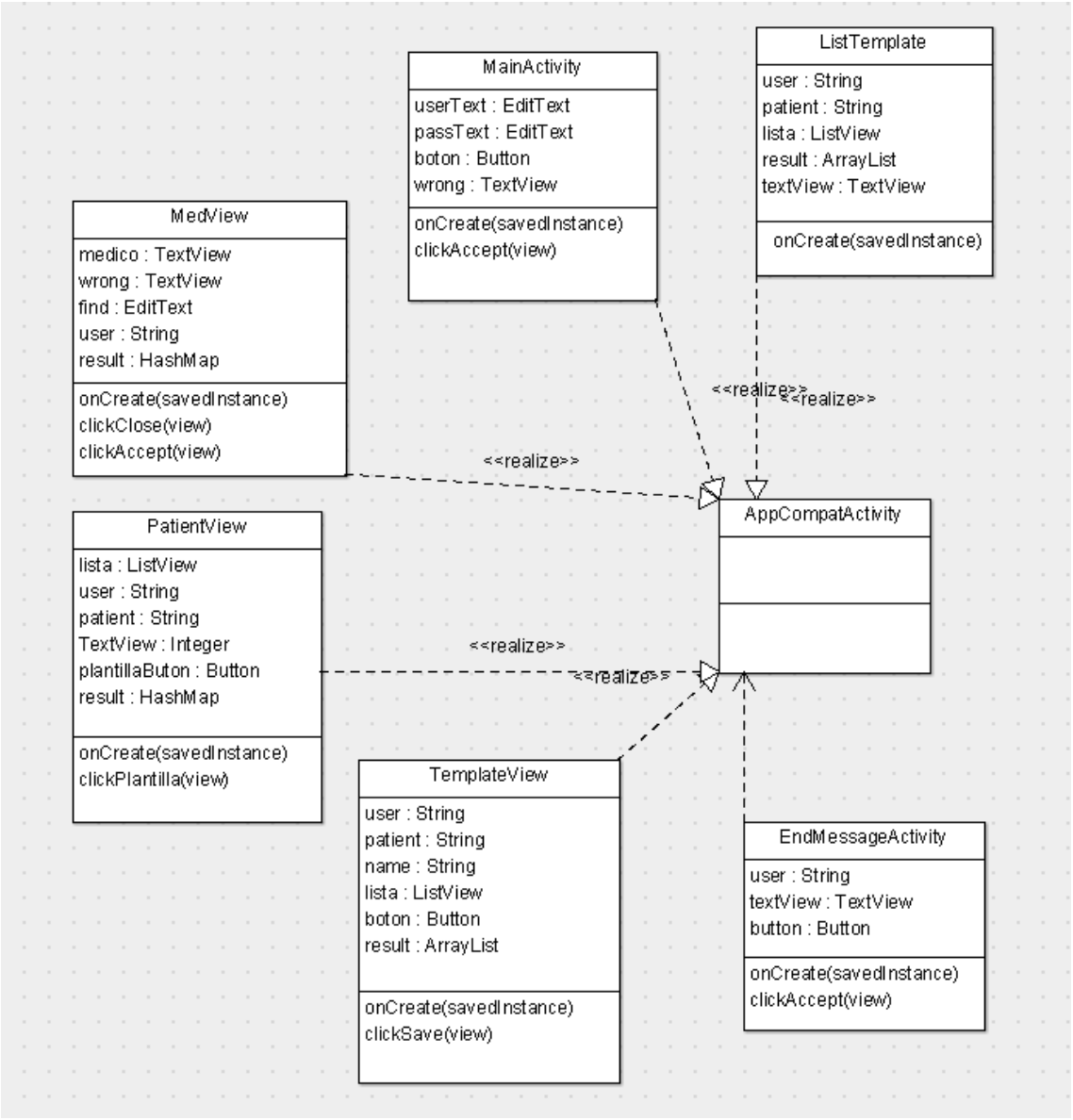

Figura 6.2.16. Diagrama de clases app móvil 2.

En este caso la clase AppCompatActivity será la clase de la extiendan todas aquellas encargadas del control de la interfaz (Figura 6.2.16).

Conclusiones y trabajo futuro.

Se ha implementado un sistema formado por una aplicación web y una aplicación móvil destinado al control de historiales clínicos de pacientes por parte de médicos.

La aplicación web ofrece la opción de insertar nuevos pacientes de dos formas, mediante el rellenado de una plantilla por defecto, o mediante la selección de una plantilla específica para dicho paciente que ha sido elaborada previamente. Ofrece también la opción de crear plantillas de test que serán realizadas desde el dispositivo móvil.

Se ha integrado además para el médico la opción de subir, borrar y descargar archivos en cada uno de los pacientes, para así, en cualquier momento tener acceso a todos los datos relacionados con dicho paciente, no solo información clínica sino documentos, de momento únicamente en formato pdf, pero será ampliable en el futuro a otro tipo de documentos más relacionados con el ámbito médico.

Por último, en la aplicación web se puede acceder al apartado de estadísticas, donde se podrán generar las mismas a partir de los test realizados.

Por su parte, la funcionalidad de la aplicación móvil permite la generación de tests por parte de los médicos a los pacientes. Estos tests serán escogidos de entre las plantillas previamente generadas en la aplicación web por el médico.

Dentro de estas mejoras se puede destacar:

- Nuevos tipos de documentos admitidos: principalmente imágenes en distintos formatos, jpeg, jpg, etc.
- Generación de recetas: permitir al médico expedir recetas online, rellenado los campos necesarios para posteriormente imprimirla y entregársela al paciente en la consulta.
- Creación de una sección para el paciente, donde pueda ver impresiones del médico.
- Un sistema de citas, juntando la parte del médico (generador de citas) y la nueva sección del paciente (observador)
- Apartado de tareas para el médico, emulando una agenda. Mediante un calendario, y estando cada día repartido en horas, para que el médico pueda listar a sus pacientes y ver a qué hora y cuantos vienen cada día, así como modificar consultas si fuera preciso.
- Control de LOPD. La LOPD (Ley Orgánica de Protección de Datos) indica el derecho al control de la documentación personal, es decir buscar un tratamiento seguro para evitar que nadie que no deba acceda a información confidencial.
- Extensión a otras plataformas, como podría ser IOS. De momento la aplicación se encuentra disponible para dispositivos móviles con sistema operativo Android. IOS está presente en aquellos dispositivos de marca Apple generalmente, ya sean móviles como Iphone o tablets como Ipad, el extender la aplicación permitiría llegar a un número más elevado de usuarios.

Conclusions and future work

We have implemented a system consisting of a web application and a mobile application. Both are for the control of medical records of patient by doctors.

The objective proposed at the beginning of the project has been reached. A clear division between the competencies of the web and mobile application.

The web application offers the option of inserting new patients in two ways, by filling in a default template, or by selecting a specific one for that patient, which one has been previously made by the doctor. It also offers the option to create test template that will be made from the mobile device.

It has also been integrated for the physician the option of uploading, deleting and downloading files in each of the patient profiles to have access at any time to all data related to that patient, not only clinical information, but document, for the moment only in pdf format, but will be expandable in the future to other types of documents related to physicians.

Finally, in the web application you can access to the statistics section, where you can generate those statics from the tests performed.

Within these improvements we can highlight:

- New types of documents supported: mainly images in different formats, jpg, jpeg, etc.
- Recipes generation: allow the physician to make prescriptions online, filling in the necessary fields to later print it and give it to the patient.
- A section for patients: where they can see impressions of the doctor.
- A dating system, gathering the doctor`s side (date generator) and the new patient section (observer).
- A task control system for the doctor, acting like an agenda: dividing every day in hours, so the physician can list their patients and what time and how many come each day.
- LOPD control.
- Extends to other platforms, such as IOS: currently the application is available for mobile devices with Android operating system. IOS is present in those Apple-branded devices generally, whether mobile as Iphone or tablets as Ipad, extending the application would reach a higher number of users.

Trabajo individual

David Cobos García

Mis aportaciones al proyecto han sido:

1. Diseño e implementación de las interfaces de la aplicación web, comportamiento full responsive, adaptando el contenido a cualquier dimensión de pantalla, desktop, tablet y móvil.
2. Introducción de fuentes vectoriales para mejor usabilidad.
3. Búsqueda e introducción de jQuery de generación de campos dinámicos en formularios.
4. Búsqueda e introducción de la librería Chart.js para generación de estadísticas (parte front-end).
5. Generación de etiquetas en la funcionalidad de las plantillas de test para poder integrar con la vista de las estadísticas.
6. Colaboración junto a mi compañero de la generación de código back-end de la aplicación web.
7. Colaboración junto a mi compañero de la construcción de estructura y código de la base de datos.
8. Generación conjunta de la estructura y escritura de la memoria.

Álvaro Alcaraz García

Mis aportaciones al trabajo de fin de grado han sido:

1. Idea de enfoque del proyecto hacia una temática médica.
2. Colaboración junto a mi compañero de la generación de código back-end de la aplicación web
3. Colaboración junto a mi compañero de la construcción de estructura y código de la base de datos
4. Construcción de base, código y diseño de la aplicación Android.
5. Conexión de la aplicación Android con la base de datos MongoDB
6. Generación conjunta de la estructura y escritura de la memoria.

Además, el código de la aplicación web se podrá encontrar en GitHub en el enlace:
https://github.com/Tocrei/patientManagerhttps://github.com/Tocrei/patientManager

Bibliografía

1.- D*escarga de Xampp:* Recuperado de

https://www.apachefriends.org/es/index.html

2.- *Descarga de Bootstrap* (diciembre de 2016). [online] Recuperado de http://getbootstrap.com

3.- *Manual oficial de Bootstrap* (diciembre 2016). Recuperado de https://librosweb.es/libro/bootstrap_3/

4.- *Fuentes de FontAwesome* [online]. Recuperado de http://fontawesome.io/

5.- *Fuentes de chart.js* [online]. Recuperado de http://www.chartjs.org/

6.- Murphey, R. (2017) *Fundamentos de JQuery* [online]. Recuperado de https://librosweb.es/libro/fundamentos_jquery/

7.-*Manuales de HTML ,CSS and PHP* [online] . Recuperado de

http://virtualtec.cl/manual-de-html5-y-css3/

8.-*Dudas y problemas similares en la comunidad informática* [online]. Recuperado de https://stackoverflow.com/

9.- PHP (2017).*La clase MongoDB* [online] Recuperado de http://php.net/manual/es/class.mongodb.php

10.- *Introducción a las Bases de Datos usando NoSQLMongoDB* - Antonio Sarasa

11.- *Descarga Android Studio* [online]. Recuperado de https://developer.android.com/studio/index.html?hl=es-419

12.- *Building Android Apps* [online] Recuperado de http://www.android-ide.com/tutorials.html

13.- *Repositorio de driver mongo para java* [online] Recuperado de https://jitpack.io/p/matfur92/mongo-java-driver

14.- Visor de imágenes médicas RadiAnt [online] Recuperado de

https://www.radiantviewer.com/

15.- Visor de imágenes OsirisX [online] Recuperado de

http://www.osirix-viewer.com/

16.- Herramienta de gestión de historiales MediConta [online] Recuperado de

http://www.infonetsoftware.com/

17.- Herramienta de gestión de historiales Doctorgest [online] Recuperado de

http://www.doctorgest.com/

18.- Herramienta de gestión de citas médicas Jagarsoft [online] Recuperado de

http://www.jagarsoft.com/agenda-medica.php

19.- Herramienta de gestión de citas médicas ClinicCloud [online] Recuperado de https://clinic-cloud.com/programa-de-citas-medicas-para-consultorio-medico-software/

20.- Manual de css [online] Recuperado de https://www.creativosonline.org/blog/guia-completa-de-css3-en-pdf.html

Anexo 1: Manual de usuario

Funcionalidades de la aplicación web:

1. Registro:

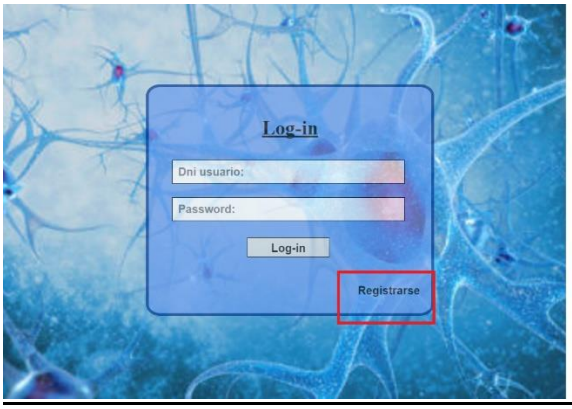

Figura A1.1. Botón de registro.

Figura A1.2. Formulario de registro.

La primera vez que se accede a la aplicación es necesario hacer un registro previo. Para ello se tiene que pulsar en "Registrarse" (Figura A.1.1) y se accederá a la pantalla que se indica a continuación (Figura A1.2).

Una vez que se han rellenado los datos, se envía una petición al administrador de la aplicación, y en caso de ser correctos, se activa la cuenta y se podrá hacer log-in (Figura A.1.2).

2. Log-in:

Una vez que el administrador ha aceptado la petición de registro, se activa la cuenta de usuario, y se podrá entrar a la interfaz de usuario previo login introduciendo el DNI y la contraseña (Figura A1.2).

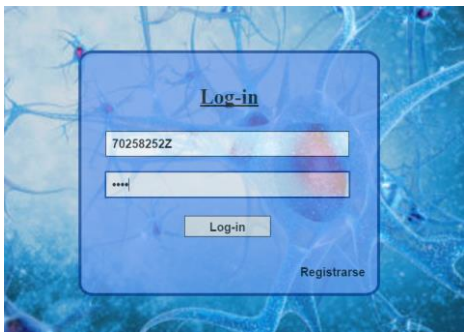

Figura A1.3. Formulario de log-in.

3. Perfil:

Para acceder al perfil personal, basta pulsar en la pestaña de "Perfil" de la esquina superior derecha (Figura A1.4), y se podrá ver la fotografía del usuario (existe una genérica para todos aquellos que entran por primera vez a la aplicación) así como sus datos (Figura A1.5).

Figura A1.4. Botón de perfil

Figura A1.5. Pantalla de perfil.

También existe un botón "Modificar datos". Si se presiona, se accederá a una pantalla en la que es posible cambiar los datos personales (en particular se puede cargar la fotografía del perfil). (Figura A1.6). Para guardar los cambios, se debe pulsar sobre el botón "Modificar datos"

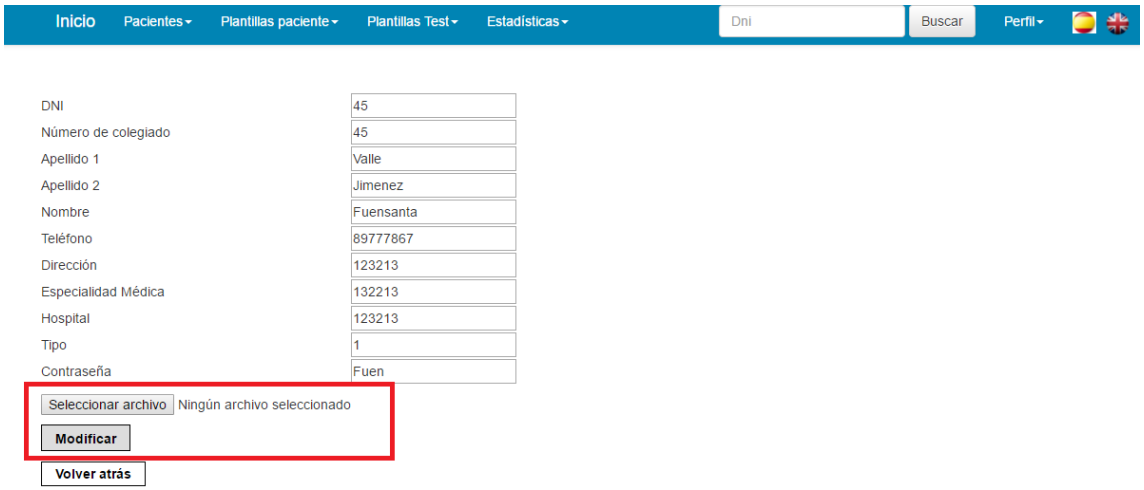

Figura A1.6. Botón de cambio de imagen.

4. Dar de alta un paciente:

Si aún no se han creado plantillas personalizadas de paciente, la única manera de dar de alta un paciente es por defecto. Para ello, puede acceder al formulario de alta desde el menú de navegación, "Pacientes -> Insertar -> Por defecto", o bien pulsando en el botón verde “Por defecto” del bloque "Dar de alta un paciente" presente en la pantalla de inicio (Figura A1.7).

Figura A1.7. Botón de alta.

Una vez ahí, se rellenarán los datos del paciente y se procederá a dar de alta presionando el botón "Dar de alta". Si se quiere cancelar el proceso, únicamente se presionará el botón "Atrás" y se volverá al menú de inicio (Figura A 1.8).

Si, por el contrario, existe alguna plantilla de paciente creada, el proceso es similar. Para acceder al alta de usuario mediante plantilla personalizada se deberá acceder desde el menú

de navegación, "Pacientes -> Insertar -> Plantilla", o bien pulsando en el botón verde “Plantilla personalizada” del bloque "Dar de alta un paciente" presente en la pantalla de inicio.

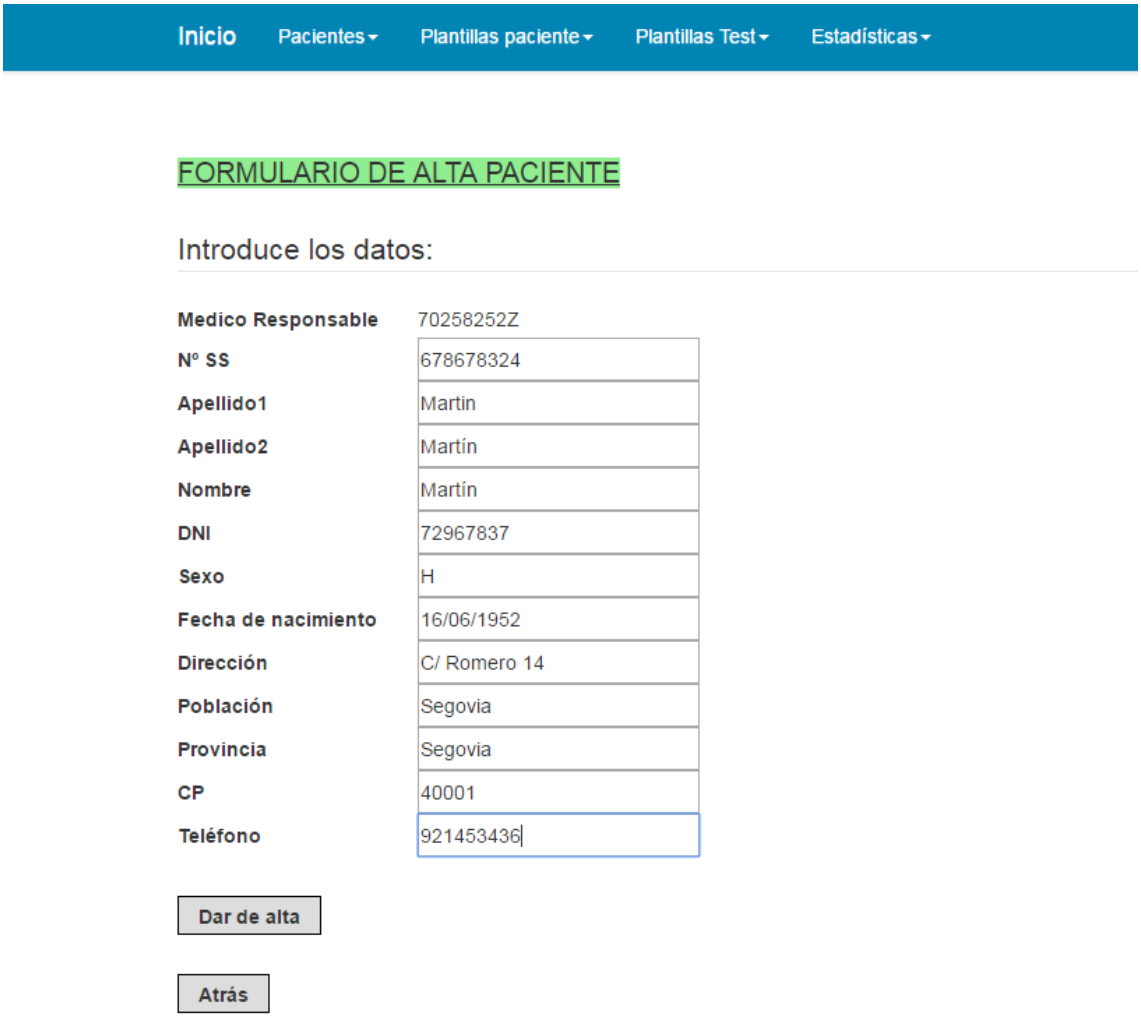

Figura A1.8. Formulario de alta.

A continuación, se muestra una lista de todas las plantillas de paciente que se hayan creado desde cada perfil. Para elegir una, simplemente se pulsará en el botón verde de la columna "Elegir" que corresponda con la plantilla buscada (Figura A1.9).

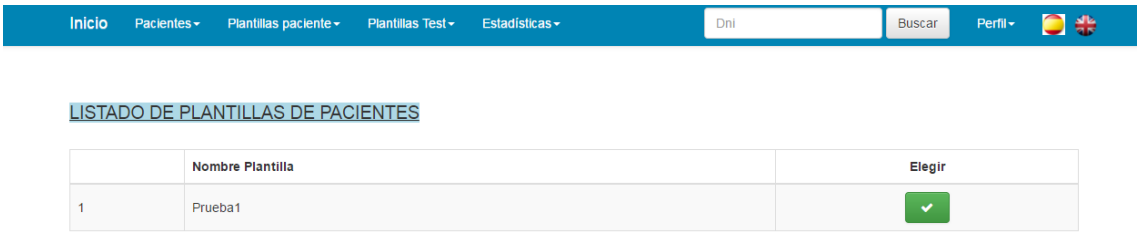

Figura A1.9. Elección de plantillas de paciente.

El siguiente paso es idéntico al de la inserción de pacientes por defecto. Se rellenarán los datos y se pulsará en el botón "Dar de alta". Para cancelar el ingreso, se pulsará en el botón "Atrás".

5. Consultar datos de un paciente:

Para consultar los datos sobre un paciente previamente registrado se podrá acceder a través del menú de navegación "Pacientes -> Consultar", o bien pulsando sobre el bloque "Consultar datos de un paciente" del menú de inicio (Figura A 1.10).

Figura A1.10. Botón de consulta.

Se accederá a un listado con todos los pacientes registrados, cada uno de los cuales tiene un asociado un botón (Figura A 1.11).

Figura A1.11. Listado de pacientes consulta.

Al pulsar sobre este botón, se accederá a la pantalla del paciente con todos sus datos, en modo solo lectura. En dicha pantalla se tienen también dos bloques en la parte derecha, donde aparecerán los tests adjuntos realizados desde la aplicación móvil y los documentos adjuntos que se hayan subido al perfil. También un icono para descargar la tabla con los datos personales a pdf (Figura A 1.12).

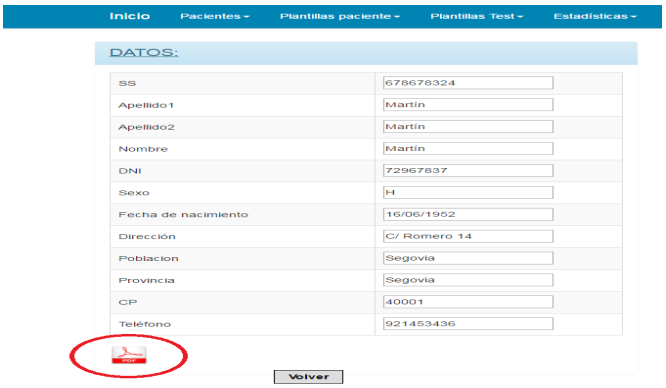

Figura A1.12. Botón descarga pdf.

Si se presiona el botón "Volver" se accederá de nuevo a la lista de pacientes.

En la imagen (Figura A.1.13) se puede observar un ejemplo de paciente con un test y un documento asociados.

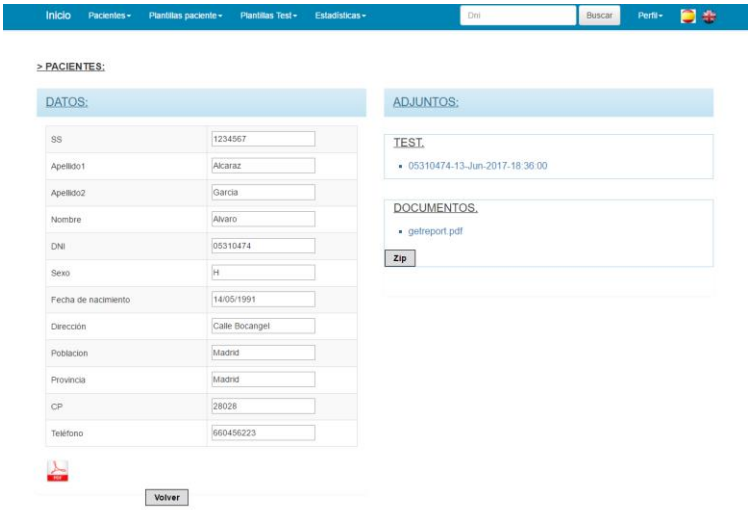

Figura A1.13. Perfil paciente con test y documento asociados

Demás aquí se puede ver el test rellenado (Figura A.1.14).

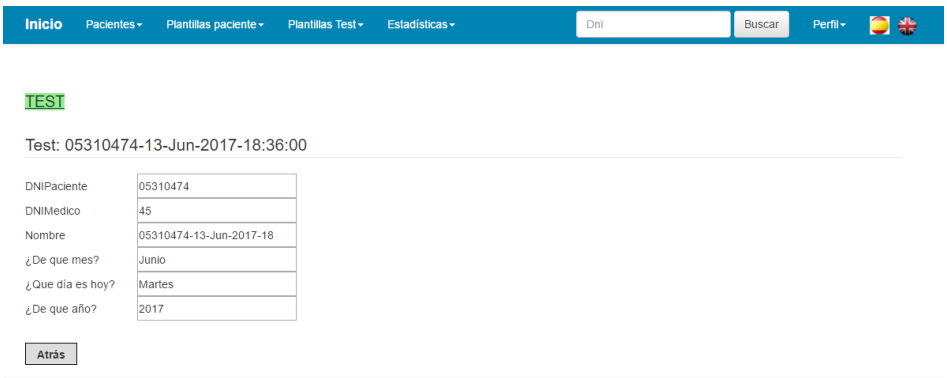

Figura A1.14. Datos rellenos de un test

6. Modificar datos de un paciente:

Para modificar los datos de un paciente, se accede a través del menú de navegación "Pacientes -> Modificar", o bien a través del bloque "Modificar datos de un paciente" del menú de inicio (Figura A 1.15).

Figura A1.15. Botón de modificación.

En la ventana generada, se muestra un listado con todos los pacientes registrados, de forma que cada uno tiene asociado un botón (Figura A1.16).

Figura A1.16. Listado de pacientes modificación.

Al pulsar sobre este botón se accederá a la pantalla del paciente con todos los datos, pero esta vez con modo escritura. Se cambiarán los datos que se necesiten y se pulsará en el botón "Modificar". Si se quiere cancelar el proceso solo hay que pulsar sobre el botón "Volver". Además, se tiene la opción de limpiar sus test adjuntos pulsando sobre el botón "Eliminar" (Figura A 1.17).

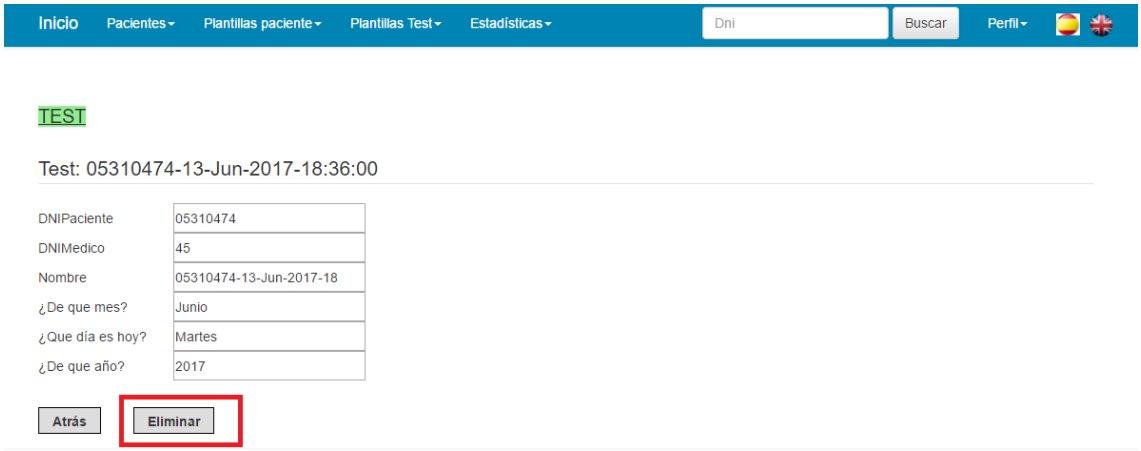

Figura A1.17. Botón de eliminar tests.

7. Dar de baja un paciente:

Para dar de baja un paciente, se puede hacer, como en todos los casos, a través del menú de navegación "Pacientes -> Eliminar", o bien pulsando sobre el bloque "Dar de baja un paciente" del menú de inicio (Figura A1.18).

Figura A1.18. Botón de baja.

Una vez hecho esto, se accederá a una pantalla con el listado de todos los pacientes dados de alta, y un botón en cada uno de ellos, para eliminar (Figura A1.19). Si se pulsa este botón el paciente se elimina automáticamente de la aplicación. Esto implica la eliminación de todos los datos relacionados entre el médico y el paciente, como pueden ser test realizados o archivos subidos al servidor.

Figura A1.19. Listado de pacientes borrado.

8. Insertar plantilla de paciente:

Para insertar una plantilla de pacientes personalizada, se puede acceder de dos maneras, mediante el menú de navegación "Plantillas paciente -> Insertar", o con su bloque correspondiente en el menú de inicio (Figura A 1.20).

Figura A1.20. Botón de insertar plantilla paciente.

Se accederá así a la plantilla de generación, siendo necesario ponerle un nombre a la plantilla, y generando los campos que se necesiten (el nombre del campo) pulsando en el botón con el símbolo "+". Una vez terminada la plantilla, se pulsará al botón "Insertar" (Figura A1.21).

Figura A1.21. Generación de campos adicionales.

Una vez hecho esto, aparecerá la nueva plantilla en el listado de plantillas de paciente. (Figura 1.22)

Figura A1.22. Nueva plantilla generada

9. Listado de plantillas de paciente:

A esta sección se podrá acceder también desde el menú de navegación "Plantillas paciente -> Listado", o desde el bloque particular en la pantalla de inicio (Figura A1.23).

Figura A1.23. Botón de listado de plantillas de paciente.

A continuación, se mostrarán todas las plantillas de paciente creadas, con tres botones cada una, "Modifica", "Más info" y "Eliminar" (Figura A 1.24).

Figura A1.24. Listado de plantillas de paciente.

La modificación no posibilita un cambio en el nombre de la plantilla, pero sí de los campos extra añadidos. Se podrán añadir más, modificar los que existan, o eliminar (para ello se dejará el campo en blanco). Cuando se finalice, se pulsará el botón "Modificar", y en caso de querer cancelar el cambio, se presionará el botón "Atrás" (Figura A1.24).

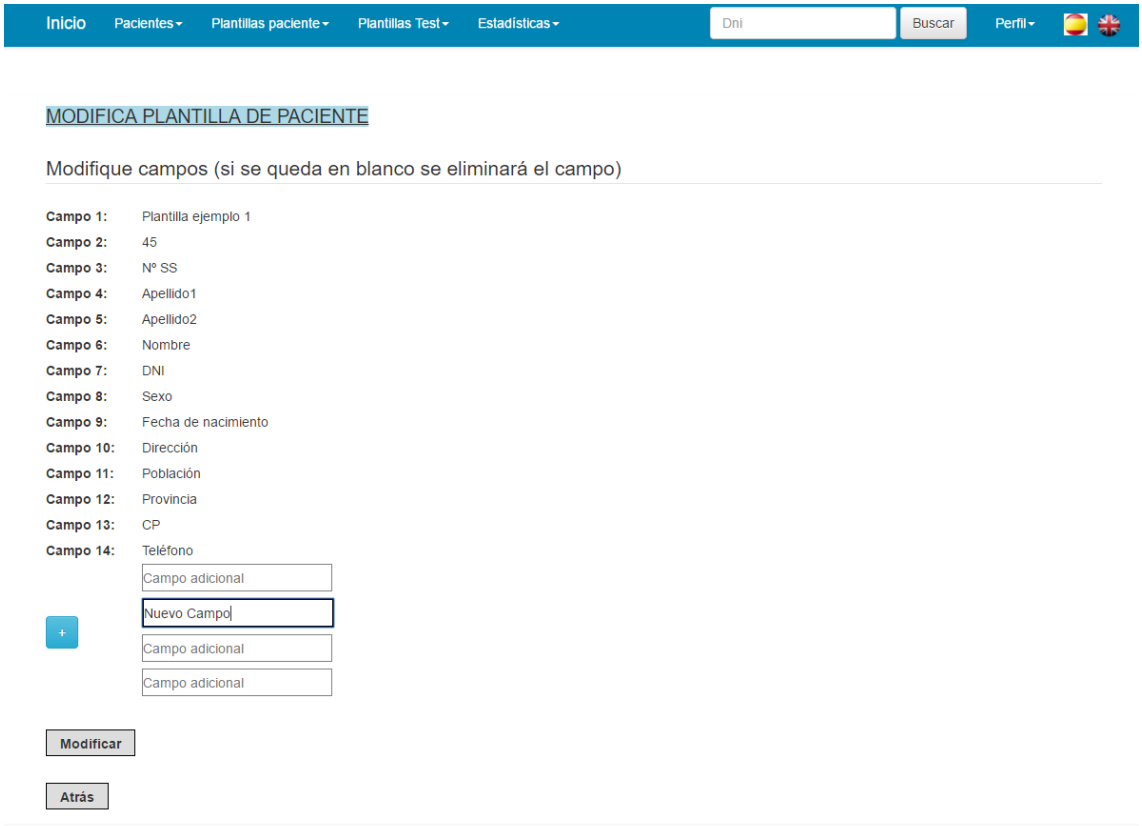

Figura A1.25. Botón de modificar campos adicionales.

Por su parte "Más info" permite acceder a los datos de la plantilla creada, sin poder cambiarlos. (Figura A1.25) Y por último si se presiona "Eliminar" en las opciones de la tabla, la plantilla se borrará y ya no se mostrará más.

10. Eliminar plantilla de paciente:

Además de seleccionando el botón “Eliminar”, (Figura A.1.24) se puede tomar un camino alternativo desde el menú de navegación "Plantillas paciente -> Eliminar" o desde su bloque en la pantalla de inicio (Figura A1.26).

Figura A1.26. Botón de eliminar plantillas de paciente.

11. Insertar plantilla de test:

Para insertar una plantilla de test personalizada, hay que acceder al formulario a través del menú de navegación "Plantillas Test ->Insertar" o desde su bloque en la pantalla de inicio (Figura A 1.27).

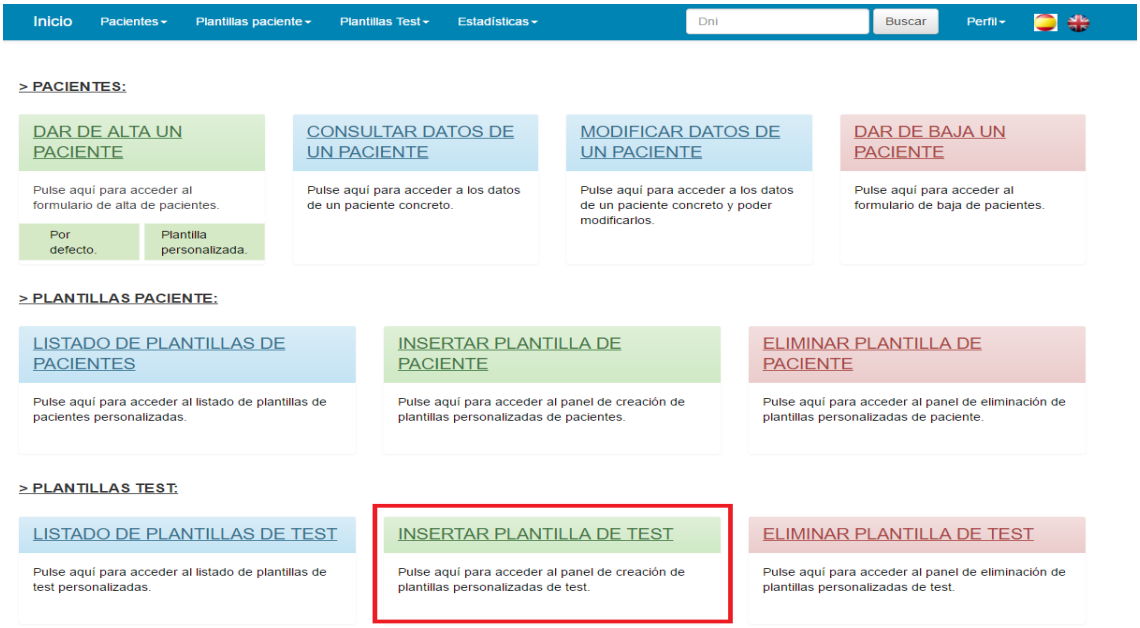

Figura A1.27. Botón de insertar plantillas de test.

Una vez dentro se podrán ver dos secciones claramente diferenciadas, una con el formulario de la creación y otra de "Etiquetas ya utilizadas" (Figura A 1.28).

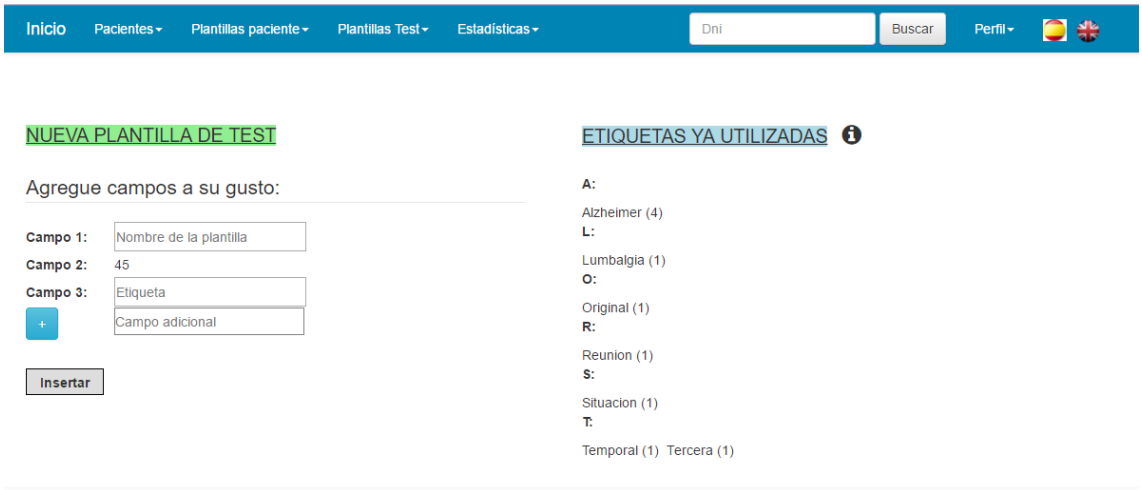

Figura A1.28. Inserción de plantillas de test.

Para insertar la plantilla de test, es necesario ponerle un nombre y asignarle una Etiqueta, que será representativa del test, por ejemplo, si es un test enfocado al alzhéimer, se pondrá "Alzheimer". Siguiendo el estilo de la creación de plantillas de paciente, se dispone de un botón "+" para generar los campos del test, que serán, por ejemplo, preguntas que se quieran hacer al paciente, y serán siempre de respuesta abierta.

El apartado de "Etiquetas utilizadas" existe para ayudar al médico a indexar su test, de cara a la creación de estadísticas. Si se pulsa sobre el título se desplegarán todas las etiquetas utilizadas en test en orden alfabético. Es recomendable revisar esto antes de crear una nueva etiqueta, para poder optimizar las estadísticas. De modo que, si la etiqueta requerida ya existe, es buena idea usarla tal y como aparece en el listado. (Figura A1.28)

Una vez que se termine el proceso de creación se pulsará en el botón "Insertar" y la plantilla quedará registrada.

12. Listado de plantillas de test:

Para listar las plantillas de tests, se accederá desde "Plantillas Test -> Listado", o desde su bloque correspondiente en el menú de inicio (Figura A1.29).

Figura A1.29. Botón de listado de plantillas de test.

En la pantalla correspondiente, aparecen listadas las plantillas de test existentes. Cada plantilla tiene asociada tres funcionalidades, "Modifica", "Más info" y "Eliminar" (Figura A 1.30). Desde el apartado “Modifica” se puede cambiar los campos del test, dejando en blanco aquellos que quieren ser eliminados (Figura A.1.31).

Figura A1.30. Opciones del listado de plantillas de test.

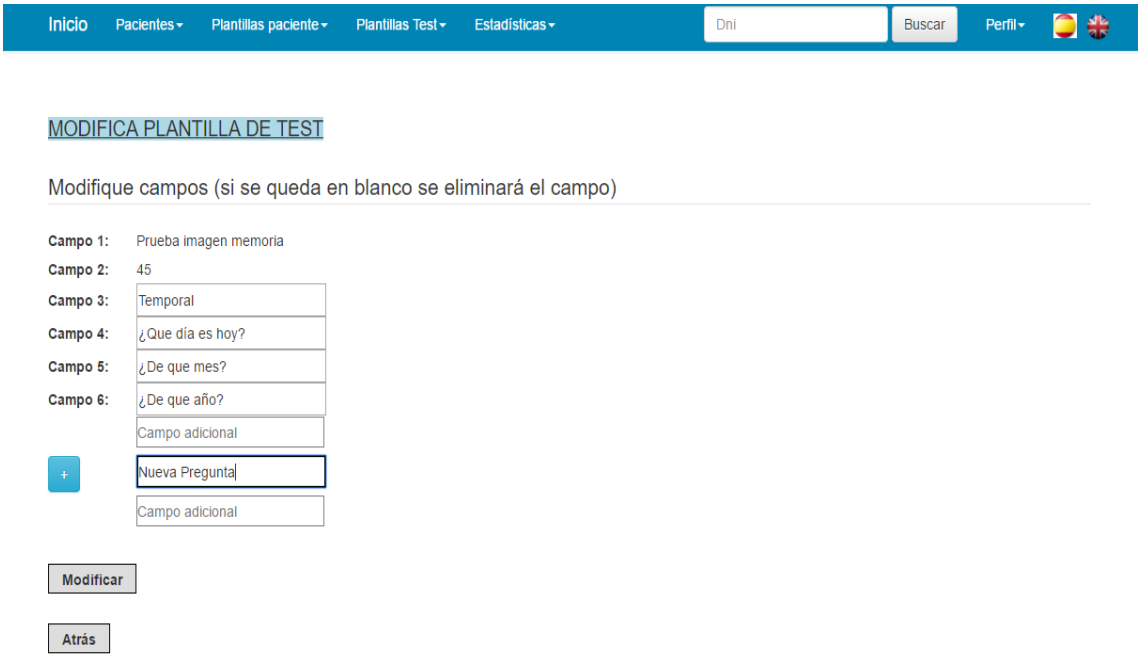

Figura A1.31. Modificación campos plantilla test.

13. Eliminar plantillas de test:

Para eliminar una plantilla de test, se accede desde el menú "Plantillas Test -> Eliminar", o bien desde el bloque correspondiente en la pantalla de inicio (Figura A 1.32).

Figura A1.32. Botón de eliminar plantillas de test.

14. Crear estadísticas:

La aplicación ofrece un modo de explotar la información generada en forma de estadísticas. Para acceder a la creación de estadísticas, se tendrá que hacer desde el menú de navegación "Estadísticas -> Crear estadística".

Una vez dentro, aparece un formulario con tres campos, los cuales se pueden rellenar o no, accediendo a múltiples tipos de estadística. El campo "Enfermedad" debe ser rellenado usando las etiquetas utilizadas en los test (de ahí que aparezca la zona de "Etiquetas ya utilizadas" para facilitar las búsquedas), el campo "Edad" debe ser rellenado con un rango de edades entre los que se quieren buscar pacientes, y por último el campo "Sexo" representa el sexo del paciente, y se debe rellenar con H para hombre y M para mujer (Figura A 1.33).

Figura A1.33. Formulario de creación de estadísticas.

Una vez se rellenen o no estos campos se pulsará en el botón "Crear" y si se quiere cancelar sobre "Volver". Si se ha seguido el primer caso, se generará un gráfico en forma de tarta, en función de los valores elegidos. Si dichos valores son los tres posibles, se mostrará un listado de todos los pacientes que cumplan los requisitos, y se podrá consultar sus perfiles.

Se debe destacar que de los 3 campos a rellena será preciso rellenar al menos uno de ellos.

En la siguiente figura (Figura A.1.34) se ve cómo sería el resultado de una búsqueda poniendo como campos la enfermedad y el rango de edad, la cual devolverá la cantidad de personas de cada sexo a las que se han realizado test con esas características.

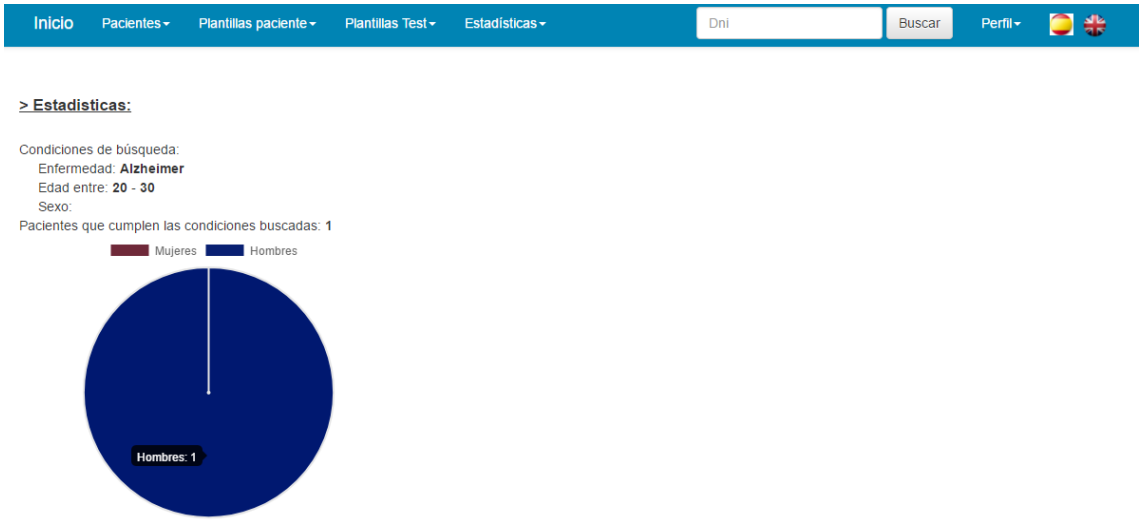

Figura A1.34. Figura de estadísticas

15. Log-out:

Una vez se ha terminado de utilizar la aplicación, por motivos de seguridad, es conveniente cerrar la cuenta de usuario. Para ello podrá pulsar en "Perfil" en la esquina superior derecha y acto seguido pulsar sobre "Log out". (Figura A1.35)

Figura A1.35. Log-out

Funcionalidades de la aplicación móvil:

Lo primero de todo es instalar la aplicación en su dispositivo móvil o tableta. Actualmente está soportado en dispositivos con versión Android 4.0 o superior.

1. Log-in:

Para utilizar la aplicación se debe haber creado una cuenta previamente en la aplicación web y que el administrador la haya aceptado. Una vez hecho esto únicamente deberá meter sus credenciales en la pantalla de inicio (Figura A1.36).

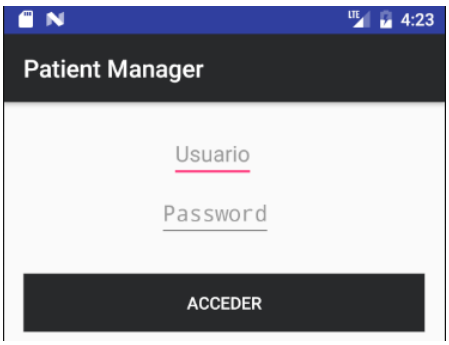

Figura A1.36. Pantalla de log-in aplicación móvil.

2. Buscar un paciente y adjuntarle un test:

Una vez identificado correctamente, deberá buscar al paciente al que desee realizar un test por su DNI (Figura A.1.37). Introduciendo un DNI de un paciente que exista y al que el médico identificado esté tratando, se verán sus datos personales y un botón para elegir la plantilla (Figura A1.38). Una vez hecho esto, se seleccionará una plantilla para realizar el test. Cabe destacar que será necesario haber realizado alguna plantilla de test previamente desde la aplicación web, sino será imposible realizar ningún test. (Figura A1.28)

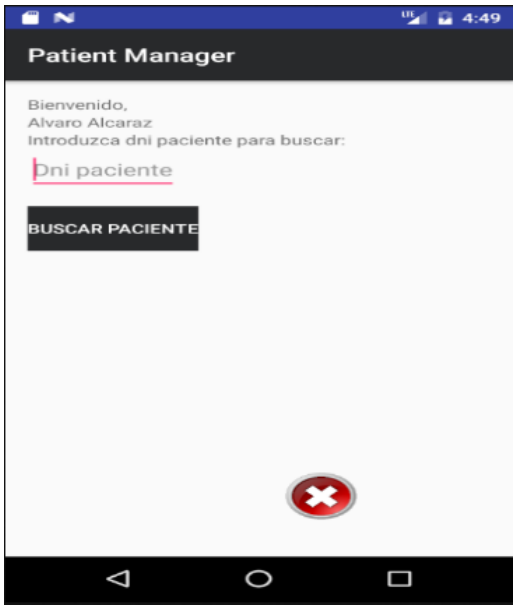

Figura A1.37. Pantalla de búsqueda de paciente

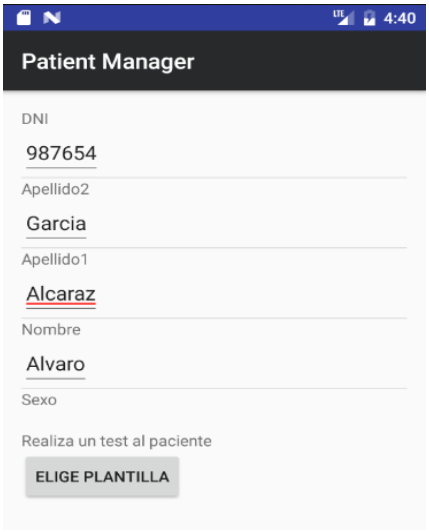

Figura A1.38. Pantalla de datos de paciente.

Por último, una vez rellenados los campos que conforman la plantilla de test escogida, se dará a finalizar test (Figura A1.39), lo que guardará automáticamente el resultado del test en la base de datos.

Figura A1.39. Pantalla de relleno de test

3. Log-out:

Para realizar un log-out , el usuario deberá encontrarse en la pantalla principal de la aplicación y pulsar sobre el icono de la X situado en la parte inferior derecha (Figura A1.40).

Figura A1.40. Pantalla de botón de log-out

Anexo 2: Manual de instalación

Para poder instalar la aplicación en entornos locales se requerirá de lo siguiente:

- Servidor independiente Apache, por ejemplo, XAMPP [1], con acceso a:
  - PHP Versión 5.6.24 o superior
  - Base de datos MongoDB Versión 1.6.8 o superior

- Todos los archivos y carpetas deben alojarse en /htdocs, directorio presente en la carpeta generada por xampp (Figura A2.1).

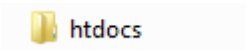

Figura A2.1. Directorio /htdocs.

- Se debe configurar correctamente el fichero "httpd.conf", alojado en el directorio xampp /apache/conf, escuchando en el puerto 80 por defecto, que se podrá cambiar por el puerto que se estime oportuno. En caso de cambiarse el puerto se deberá apuntar a la ruta del proyecto desde localhost:(nuevo puerto) y reiniciar xampp desde el panel de control (Figura A2.2).

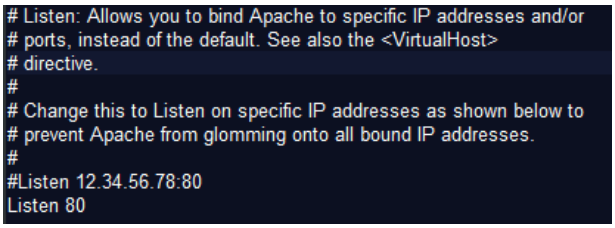

Figura A2.2. Configuración de puerto en xampp.

- Se debe tener lista la base de datos no relacional MongoDB, para lo cual se debe ejecutar mongod.exe desde la consola de comandos para que escuche, y mongo.exe para entrar en nuestra base de datos. Una vez ahí, y con xampp ejecutando su apartado de Apache, se introducirá la dirección de la web en la barra de navegación de cualquier navegador (Figura A2.3).

Figura A 2.3. Comandos de ejecución de servidor y cliente en consola en MongoDB.

- NOTA: el directorio xampp debe instalarse en una ruta que tenga permisos de escritura, lectura y ejecución hacia abajo, es decir acceso a todas las subcarpetas del mismo. No importan estos permisos hacia fuera.

Para poder instalar la aplicación sobre un servidor:

- Los requisitos de versiones son los mismos.
- Es necesario que todo el contenido que para local se insertaba en la carpeta /htdocs esté alojado en la raíz del servidor.
- Además, se necesitará dar una serie de permisos especiales para MongoDB, por razones de seguridad:
  - **$conexion = new MongoClient("mongodb://UsuarioBD:PasswordBD@paginaWebAlojada:Puerto/NombreDeLaCarpeta?authSource=admin");**

```
$conn = new MongoClient("mongodb://davidalvaro:patientmanager2017@127.0.0.1/patientmanager?authSource=admin");
$db = $conn->patientmanager;
```

Figura A 2.4. Código para asegurar la conexión al servidor web

Para la instalación de la aplicación móvil.

- De nuevo se necesitará un servidor independiente Apache con:
  - Base de datos MongoDB Versión 1.6.8 o superior.
- Para poder generar el apk que se va a instalar en el móvil es necesario disponer del programa Android Studio, versión 2.2.3 o superior.
- Una vez abierto el programa, al igual que con la aplicación web, la conexión con el servidor deber ser segura, otorgando unos permisos especiales, donde habrá que poner:
  - Nombre de la base de datos
  - Contraseña de la base de datos
  - Nombre del usuario de la base de datos
  - Página de servidor donde está alojada.
  - Puerto de acceso
  - Carpeta donde se encuentra la base de datos en el servidor
- Una vez controlado todo esto, se buscará en el proyecto los ficheros que realizan las conexiones a la base de datos y se cambiará la siguiente línea con los datos anteriormente guardados.
  - `MongoClientURIuri = new MongoClientURI( "mongodb://UsuarioBD:PasswordBD@paginaWebAlojada:Puerto/NombreDeLaCarpeta?authSource=admin" );`

```
HashMap<String, Object> mapa = new HashMap<String, Object>();
MongoClientURI uri = new MongoClientURI( "mongodb://davidalvaro:patientmanager2017@127.0.0.1/patientmanager?authSource=admin" );
MongoClient mongoClient = new MongoClient(uri);
MongoDatabase prueba = mongoClient.getDatabase("patientmanager");
```

Figura A 2.5. Código Android para asegurar la conexión al servidor

- Si todo funciona correctamente al compilar la aplicación debería generarnos un apk, el cual habría que instalar en un dispositivo Android y estaría listo para su uso.

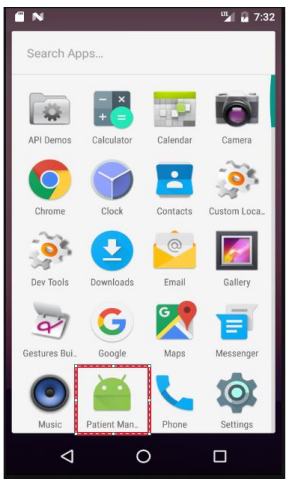

Figura A 2.6. Imagen de Apk instalada en un dispositivo Android